THE BEST SELLING COMPUTER PROJECTS MAGAZINE 65C02 CMOS TYPE 6502 JULY 1984

14 INFRA RED LINK
24 MICRO CONTROLLER 29 MICROPROC
31 EPROM REF JUNE ISSUE
VIA 6522 51
ROBOT 63 TO 71
BOOK LIST
72 BOOK REVIEW
74 DISC
85 LIGHT PEN REF
SEE ↑ P24

ELECTRONICS & COMPUTING

MONTHLY

AN EMAP PUBLICATION

USA $2.95
Germany D5.80 90p

WIN
a computer holiday for two

LIGHT CONTROL
Link up your micro with infra-red

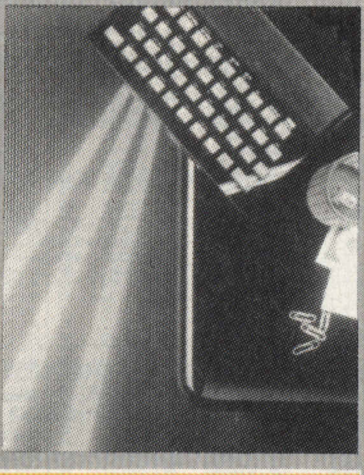
DISK DRIVE SPECIAL-SURVEY+BBC UTILITY

PROGRAM POWER WITH 3D GRAPHICS·EXPLORING OS9

MSX INVASION·BUILD A LEGO ROBOT

ELECTRONICS & COMPUTING Contents

Vol. 4 Issue 7



COVER PHOTO BY ROB BRIMSON

Electronics & Computing Monthly
Scriptor Court, 155 Farringdon Road,
London, EC1R 3AD

Editorial 01-833-0846
Editor Gary Evans
Assistant Editor William Owen
Production Editor Liz Gregory
Administration Serena Hadley

Advertising 01-833-0531
Advertisement Manager Anthony Herman
Chief Executives Richard Jansz
Francis Lee
Classified Martin Derx

Production 01-833-0531
Art Editor Jeremy Webb
Make-up Time Graphics
Publisher Alfred Rolington

Distribution
EMAP National Publications

Published by
EMAP Business and
Computer Publications

Printed by
Riverside Press, England

Subscriptions
Electronics & Computing Monthly,
(Subscription Department),
Competition House, Farndon Road,
Market Harborough, Leicestershire.

Electronics & Computing Monthly is normally published on the 13th day of each month.

 Subscription rates: UK £10.70 incl. post. For overseas rates apply to Subscription Dept., Competition House, Farndon Road, Market Harborough, Leicestershire. Back issues available from EMAP National Publications (*E&CM* Back Numbers), Bretton Court, Peterborough PE3 8DZ. Phone: 0733 264666

ABC
MEMBER OF THE AUDIT BUREAU OF CIRCULATIONS

PROJECTS

FEATURES

REVIEWS

PLUS

And within *Your Robot*

A report on the recent USA robotics conference, motors explained, a DIY robot and reviews of the latest books on this fascinating aspect of computing.

Acorn gets down to business

Long overdue, Acorn's (second) second processor, a Z80 based device, is now available.

As Acorn rightly claim, with a Z80 processor running CP/M, the BBC has now become a viable business machine in its own right, and one of the fastest systems on the market for something less than the usual price (on a quick calculation, £1,300 for a full system including disk drives, printer, monitor, terminal and processor).

The second processor consists of a Z80B which, running at 6MHz, is one of the fastest Z80s available; and 64K of RAM of which 55K is available when running under CP/M2.2.

Data is exchanged via the BBC tube interface. This gives dual operation: the Z80 controls the application program, and the BBC's 6502 the I/O, screen and system routines.

The Z80 runs true CP/M2.2, which conforms to industry standard but allows full access to the BBC micro's own machine OS. Programs can use the GSX graphics extension to CP/M.

The great advantage of CP/M is, of course, the enormous range of software available. A full suite of applications software is available free with the Z80 second processor. Included is three office productivity programs: MemoPlan (wordprocessor); FilePlan (database); and GraphPlan (spreadsheet with graphics). There is also an integrated accounting system; the Nucleus System Generator (1984 RITA software product of the year); a selection of additional programming languages (CIS COBOL, professional BASIC and BBC BASIC) and of course the CP/M2.2 (with GSX graphics).

With the Z80, this extensive range of software should make the BBC a very attractive, and perhaps truly professional product.

MEMEX

At the time of closing for press we have still not managed to resolve the problems surrounding the publication of the second part of our Memex project. Negotiations aimed at resolving the situation are however at an advanced stage and we expect to be able to publish the article in one form or another in our next issue.

Spoilt for choice

The launch of British versions of MSX micros will mean that for the first time computers with such familiar brand names as Sony and Hitachi should be on sale in time for this year's Christmas sales bonanza. Add to this the facts that Commodore C16, a machine designed to rival the Spectrum at the lower end of the market, was given its first UK showing at the Fifth *International Commodore Computer Show, plus the recent* introduction of the Amstrad CPC464 and Tatung Einstein, and it will be quite apparent that there will be no shortage of choice for the consumer in the latter half of this year.

Predicting which of the many models will grab the lion's share of sales at this stage would indeed by a foolhardy exercise. However, if last year is to be taken as a guide, it will be the companies that manage to get their product onto the retailer's shelves that will clean up in terms of sales. To this end Commodore with a possible package deal on the CBM64 must be in a good position and if Amstrad's production targets are met, they should also be in with a chance of making the top five in terms of machines sold in the last quarter of the year.

The Spectrum too is likely to achieve a place in the best sellers list particularly as both versions of the computer look ready for a price cut. It's still too early to predict the performance of MSX standard machines as this will very much depend on the various pricing policies adopted by members of the MSX group. To date no specific prices have been announced for these machines but prices of around £200 for a 64K machine have been floated. At this price though, the Amstrad machine with its green screen monitor and built-in data recorder will seem a far better buy.

All in all, it looks like being a fairly cut throat market toward the end of the year and one that is likely to see a number of manufacturers shaving their margins to the bone.

First past the post

Between them the Z80 and the 6502 dominate the UK and USA microcomputer market. The reason for this is almost entirely due to the fact that these processors were at the heart of the first machines that could truly be called home micro computers. These were, for those of you whose memory's do not reach that far back, the original PET and Apple (6502), and the first in the family of TRS 80 computers (Z80). These machines set the pattern for those that followed with the result that it is rare for an eight bit machine aimed at the home market to opt for any other MPU.

The great shame about this state of affairs is that the 6809 processor, which is by common consent the most versatile and powerful of eight bit MPUs, has been largely overlooked. The sole exception in this country is the Dragon together with its clone the Tandy Colour Computer.

Those users restricting themselves to BASIC programming will not be disadvantaged but anyone experimenting with machine code programming will lose out on the elegant nature of the 6809 architecture and its unique attributes such as the ease with which Position Independent Code can be implemented.

It's too late for the 6809 to achieve any significant presence in the UK, and the processor is destined to go down in history as one of the "also rans". It's not too late though for anybody buying a machine with a view to developing machine code programming skills to opt for the Dragon computer, the only problem is that the experience will make writing code for any other processor seem an uphill task.

Pirates strung up by dongle

Games pirates may rue the day of the dongle if other software houses follow the lead of Microdeal, by providing a security key with each computer game sold. Games purchasers may not be pleased either, as the devices will push up the price of a game by £1.50.

Microdeal are introducing 40 new games, each with a security key, under the name Tom Mix Software. Each key contains two TTL ICs, but the manufacturers are understandably reluctant to give more information than that about the composition.

Presumably the devices can be active, using the 5V supply line of the RS232 interface, and the joystick port can be read directly by machine code. Whatever the case, Microdeal defy anyone to plough their way through the lump of plastic and resin to either break the key or construct their own.

QL owners frustrated

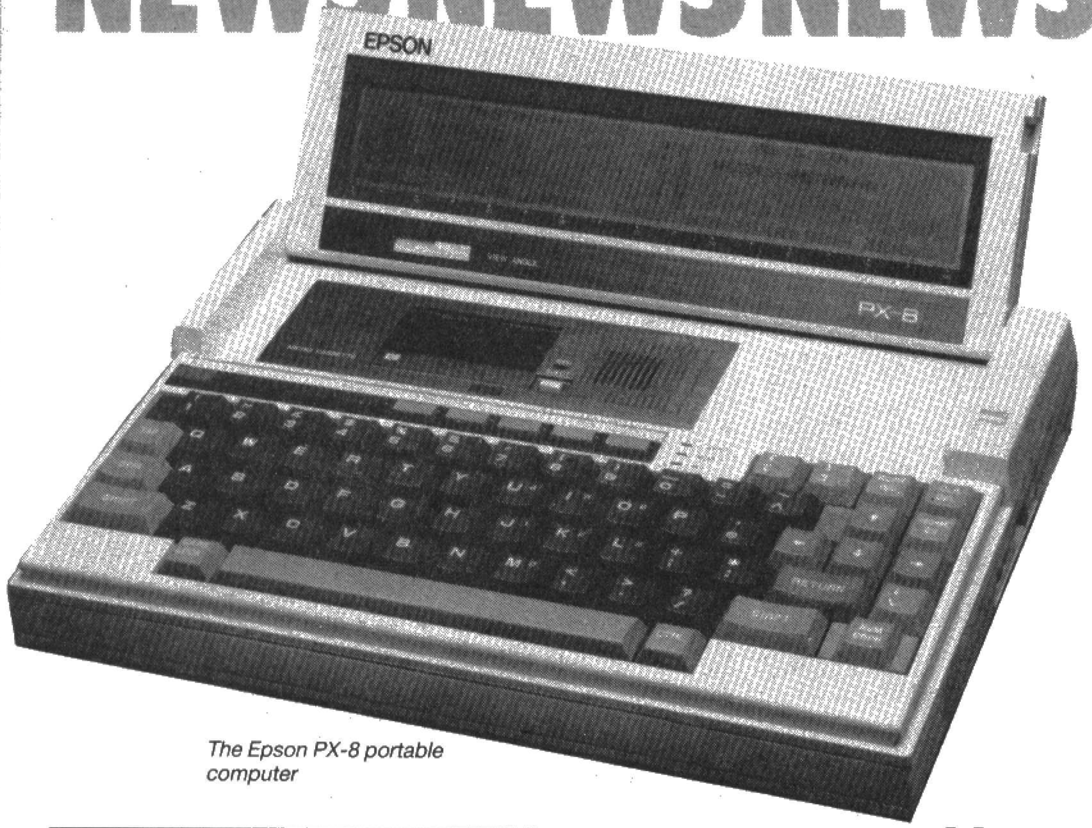
Prospective QL owners might be interested in the latest piece of QL gossip: that machines are being sent out by Sinclair minus the SuperBASIC programming section of the manual and, in some cases, minus leads!

Little letters are hidden away in the box promising delivery of the missing parts 'within 14 days'. Needless to say our contacts have not yet received the absent goods and the 14 day deadline is long gone.

PS. Next month we will have news of lots of lovely QL bugs, of both the hardware and software variety. Watch this space.

Eagle-eyed bargain hunters will have noticed a particularly attractive deal in last month's *E&CM*. The Twillstar advert on page 25 featured an EPROM programmer for only £5. Now we've heard of shaving margins to the bone but £5 is rather too good to be true. The price for the programmer should have been shown as £59. The error occured because, and here we have to introduce a technical printing term, the 9 dropped off our artwork.

Our apologies to Twillstar for any inconvenience the error may have caused.



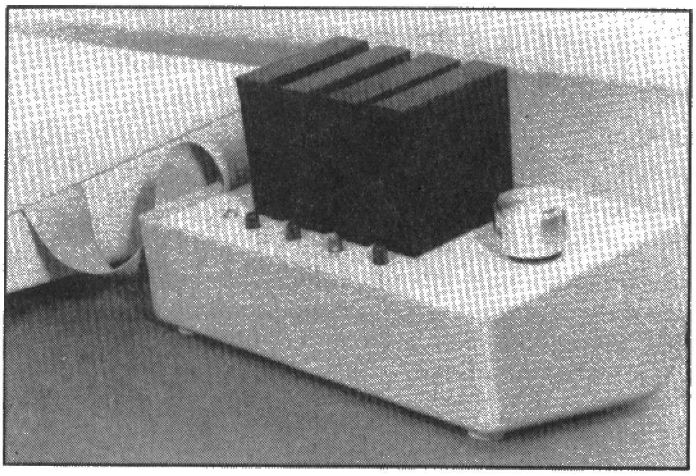
The Epson PX-8 portable computer

Nine machines – one standard

At a recent press briefing, the principle members of the MSX group gathered together to sing the praises of their standard for microcomputers. The occasion turned out to be rather a non-event as none of the manufacturers gathered under the MSX banner were prepared to make any announcements as to price or delivery of their incarnations of the MSX formula. Some hardware was on display but the computers were all models designed for the Japanese market and nobody was saying how many changes would be incorporated in the versions due for the UK.

The CETEX trade show a few days later saw some of the MSX licencees displaying their wares but firm commitments to price and delivery were still not forthcoming.

For a fuller account of the MSX standard and a look at some typical hardware see our MSX report elsewhere in this issue.



Pictured above is one of a range of BBC ROM extensions from Ramamp computers. The ROM extension unit accepts up to four ROMs. These are inserted into card sockets which are then plugged into the board (price £45.80). Also available is a ROM extension board (6 extra slots, price £26.80). A RAM-ROM board (price £33.50) and an intelligent EPROM programmer (£78.00). *Ramamp Computers, 25 Avon Drive, Whetstone, Leicester.*

PX portable

Epson, the company who broke new ground in 1982 with the first truly portable business micro, the HX20, have replaced that machine with the PX-8, an A4 sized machine weighing less than four pounds which runs the Digital Research CP/M operating system.

The PX-8 uses a flip-up LCD to give a larger screen format than the HX20, of 80 characters by 40 lines. The PX-8 includes a full range of built-in business software on micro-cassette: WordStar, Calc, Scheduler and address file.

The portable offers up to 184K RAM with 64K as standard. Interfaces include A/D, RS232C, bar code reader and universal expansion port. The PX-8 is priced in the affordable sub-£1000 bracket at £798 for the standard machine and £998 with RAM pack extension.

Where to get your Interbeeb

Following our review of DCP Microdevelopment's in last month's issue of **Your Robot** we have been asked to point out that any company interested in distributing the device should contact DCP direct at 2 Station Close, Lingwood, Norwich, NR13 4AX.

Technical enquiries should be directed to Cambridge 833902 while production enquiries will be dealt with on Hemel Hempstead 64225.

New heights of realism for games players

The potential of home computers for the application of sophisticated (and not a few crude) games of strategy, action and real time simulation is beginning to be realised in some games now coming onto the market.

Ant Attack was the first to break new ground with four different perspectives and true 3D in an arcade game. In the field of simulation, Acornsoft's recent release, Aviator, rivals the IBM PC version of an aircraft simulator, with stunningly real handling and speed of a spitfire flying over a 3D (black and white) wire graphics landscape.

Two products released in the last week have overcome the boredom of text-only adventure, but each from a different approach, and have taken strategy into the realms of arcade action.

The first, Mugsy Hits Town, from Melbourne House, is essentially a text based adventure game, with a difference.

Mugsy is claimed to be 'the first interactive computer comic strip'. Mugsy is our hero, a gangster from the East Side who shure don't shpeak too well. His one aim is to be the baddest cat in town; to do this, he gotta boy de mob's artillery and ammo, organise de protection rackets, bribe da cops, and most important of all, he gotta make a lotta dough – before Rocco puts out a contract on him.

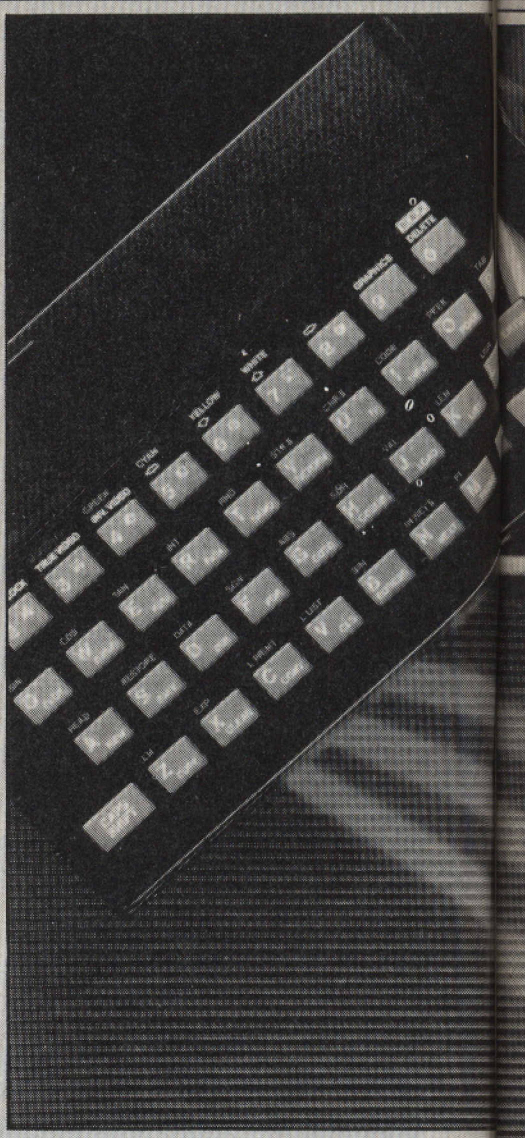
Mugsy Hits Town uses excellent cartoon graphics drawn by Russell Comte using the Melbourne Draw graphics utility. Decisions have to be made in the standard adventure games fashion, but at certain stages of the game fire power is all important.

Beyond Software's new game for the Spectrum takes the unity of adventure, strategy and arcade action one step further. Psytron has a total of six highly detailed screens with different action occurring in each one, that is, real-time arcade action. But at the same time the player must control a complex range of resources in order to maintain his defences.

The possibility for serious application of such a level of sophistication in education is quite apparent. Accurate simulation of a variety of situations, actions, and decision making processes is now within the grasp of home micros, as well as mainframes and minis. All we have to do now is remember that in real life the smash ups and tears are real.

MESSAGES by infra-red

Wires get tangled, light doesn't. Using this infra-red transmitter/receiver you can link a keyboard to a computer, a computer to a robot, a control system to a . . .



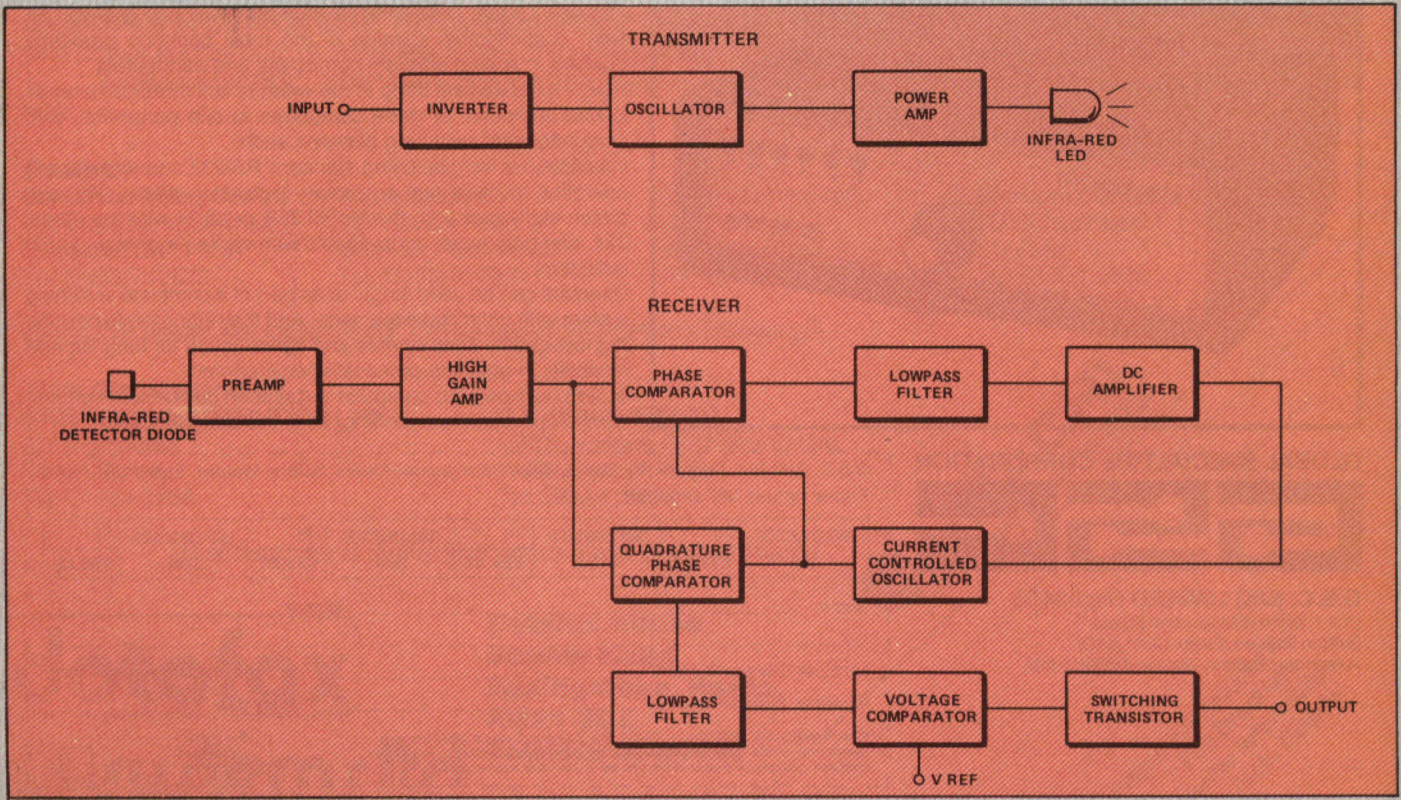
This data link is designed to handle serial data at baud rates of up to about 300. The data is carried by a modulated infra-red beam so that no connecting wires between the transmitter and receiver circuits are necessary. Possible applications for the unit might be within a "wireless" keyboard or controller for a computer system, or to provide the link between a computer and some form of robot. A lack of connecting wires could be especially useful in the latter application.

The receiver accepts ordinary TTL type (nominal 0 to 5 volt signals), and the transmitter provides an output of the same type. Both circuits require a single 5 volt supply which can conveniently be provided by four NiCad cells wired in series if battery operation of either unit is needed. The system works well and with few directivity problems at distances of up to about 1 to 2 metres. It will operate over an absolute maximum distance of approximately 6 to 7 metres, but alignment of the transmitter and receiver diodes will inevitably become much more critical. Also, due to the high gain of the receiver, it is essential to ensure a low level of stray pick-up from any other equipment used in the overall system.

Keyed operation

Data links often make use of frequency shift keying (fsk), where the transmitter provides two output frequencies (one to

Figure 1. Block diagram of infra-red data link.



represent a low logic level, and the other to represent a high logic level). This equipment uses a somewhat simpler system where one logic level is represented by an output frequency, but the other is simply represented by no signal at all. This system has been adopted due to the use of an NE567 phase locked loop (PLL) tone decoder as the basis of the receiver circuit. The block diagram of **Figure 1** shows the make up of the system.

Of the two sections of the equipment the transmitter is the more simple, and is basically just an audio oscillator running at a frequency of several kilohertz. The input signal is used to key the oscillator on and off via an Inverter stage, and this stage is merely needed to avoid having a phase inversion through the overall system. The output current capability of the oscillator is inadequate for driving the infra-red LED properly, therefore a simple power amplifier is used to give a greater drive current.

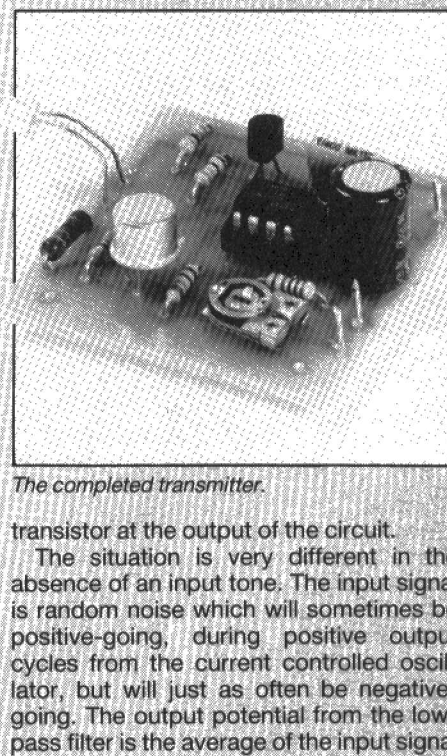
The output from an infra-red LED is not very strong, and the signal voltage produced at the receiving diode is extremely small. A simple DC link would be unusable at a range of more than a few millimetres since, at greater ranges, the signal from the transmitter would be swamped by noise, and things like temperature drift could also make such a system unusable.

The use of a pulsed infra-red beam is much more practical since high gain audio amplifiers at the receiver can be used to boost the signal up to the level required to drive some form of tone detector. This avoids problems with temperature drift, and as the tone signal is less affected by noise problems, a much greater range can be obtained. The main drawback of this method is that the maximum data rate which can be handled is substantially less than a simple DC system, but this is a price that has to be paid to obtain a useful maximum range.

The receiver has a single stage preamplifier followed by a two stage high gain audio amplifier. The seven other stages of the unit are all provided by the NE567 PLL tone decoder plus a few discrete components. A fairly conventional phase locked loop circuit is used in the NE567, with the input signal and the output of a current controlled oscillator being fed to the two inputs of a phase comparator. The output signal from the phase comparator is filtered and amplified to give a DC control signal for the current controlled oscillator. This arrangement locks the oscillator onto the same frequency as the input signal, and also keeps it in phase with the input. If it should lag behind the input signal, then the output voltage from the phase comparator, filter, and amplifier circuits rises and gives a stronger control current to the oscillator. This boosts its operating frequency to bring it in line with the input signal. Conversely, if the oscillator should run ahead of the input signal, the control current to the oscillator is reduced, and its frequency and phase are brought back in line with the input signal.

Normally, in a tone decoder application, it is the output voltage from the phase comparator, filter, and amplifier combination that is of interest. This rises and falls in sympathy with the input frequency, and, using a two-tone fsk input signal this gives two output voltages. It is quite easy to process this signal to give standard logic level outputs. The NE567 uses a different and unconventional approach, but one that nevertheless seems to work extremely well in practice.

Some of the input signal is fed to a quadrature phase comparator where the phase of this signal is compared with that of the current controlled oscillator. The quadrature phase comparator is really a form of electronic switch, and it effectively allows the input signal to flow through to the output during positive output half cycles from the current controlled oscillator. When an input tone is present, it results in a series of positive output pulses with the input signal half wave rectified. These pulses are smoothed by a lowpass filter circuit to give a strong positive DC signal. This is fed to one input of a voltage comparator, while the other input is fed with a (lower) reference voltage. This sends the output of the comparator high and activates a switching transistor at the output of the circuit.



The completed transmitter.

The situation is very different in the absence of an input tone. The input signal is random noise which will sometimes be positive-going, during positive output cycles from the current controlled oscillator, but will just as often be negative-going. The output potential from the lowpass filter is the average of the input signal voltage, which in this case is roughly zero. This sends the output of the comparator low and switches off the output transistor.

Thus the system gives what is effectively a DC coupling from the input of the transmitter to the output of the receiver.

Transmitter circuit

The transmitter is built around the ever useful 555 timer device, and the full transmitter circuit appears in **Figure 2.**

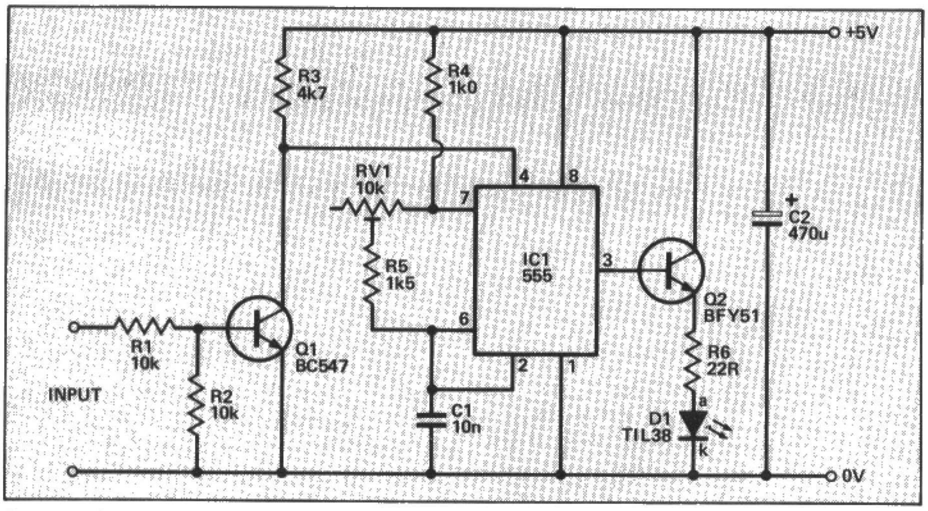


Figure 2. Transmitter circuit diagram.

IC1 is a standard 555 astable oscillator circuit with an adjustable output frequency. VR1 is trimmed to bring the output frequency within the narrow lock range of the receiver. The inverter stage is based on TR1, and with a low input level this gives a high control signal to pin 4 of IC1 and enables oscillation. With a high input level, pin 4 is taken low, oscillation ceases, and IC1's output goes low. TR2 is an emitter follower buffer stage which enables the fairly high drive current of about 100 milliamps for D1 to be readily achieved. R6 is a current limiting resistor which controls the LED current. The specified value of 22R gives about the highest safe current using a single LED, and a lower value should not be used. With the oscillator diabled and the output of IC1 in the low state D1 is switched off and no significant LED current flows.

The current consumption of the transmitter is about 55 milliamps.

Receiver circuit

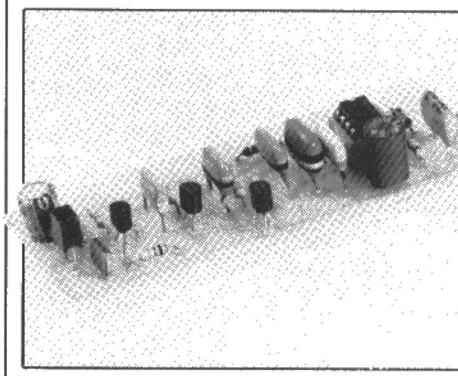
Figure 3 shows the circuit diagram for the receiver unit.

D2 is the infra-red detector diode, and this is a sensitive device which has an integral infra-red filter to reduce interference from visible light sources. It is connected so that it is reverse biased by R7, and the leakage current of the device depends on the infra-red level to which it is subjected (increased level giving increased leakage). Therefore the pulses of infra-red from the transmitter give small negative output pulses from the junction of D2 and R7.

TR3 is used as a common emitter amplifier, and R10 is used to give local negative feedback which reduces the otherwise excessive voltage gain of this stage. The amplified signal is coupled by C5 to two common emitter amplifiers which provide most of the circuit's voltage gain. The values of the coupling capacitors have been made as small as possible and are consistent with efficient coupling at the frequency of the input signal. This severely attenuates 100 Hertz mains hum which is generated by mains powered lighting, and which might otherwise greatly reduce the efficiency of the system.

IC2 is the NE567 phase locked loop tone decoder. R16 and C8 are the timing components for the current controlled oscillator. C9 is the capacitor in the low-pass filter at the output of the phase comparator, and this operates in conjuction with an internal resistor of IC2. C10 is the smoothing capacitor at the output of the quadrature phase comparator. The internal switching transistor at the output of IC2 has an open collector output, and R17 is the discrete load resistor for this.

The receiver has a current consumption of about 10 milliamps with no Input tone,



The completed receiver.

and around 25 milliamps when locked onto the signal from the transmitter.

Construction

Both circuits are built on small printed circuit boards, and these are illustrated in **Figure 4** (transmitter) and **Figure 5** (receiver). There is nothing unusual in the construction of either board, and there should be no difficulty in building them.

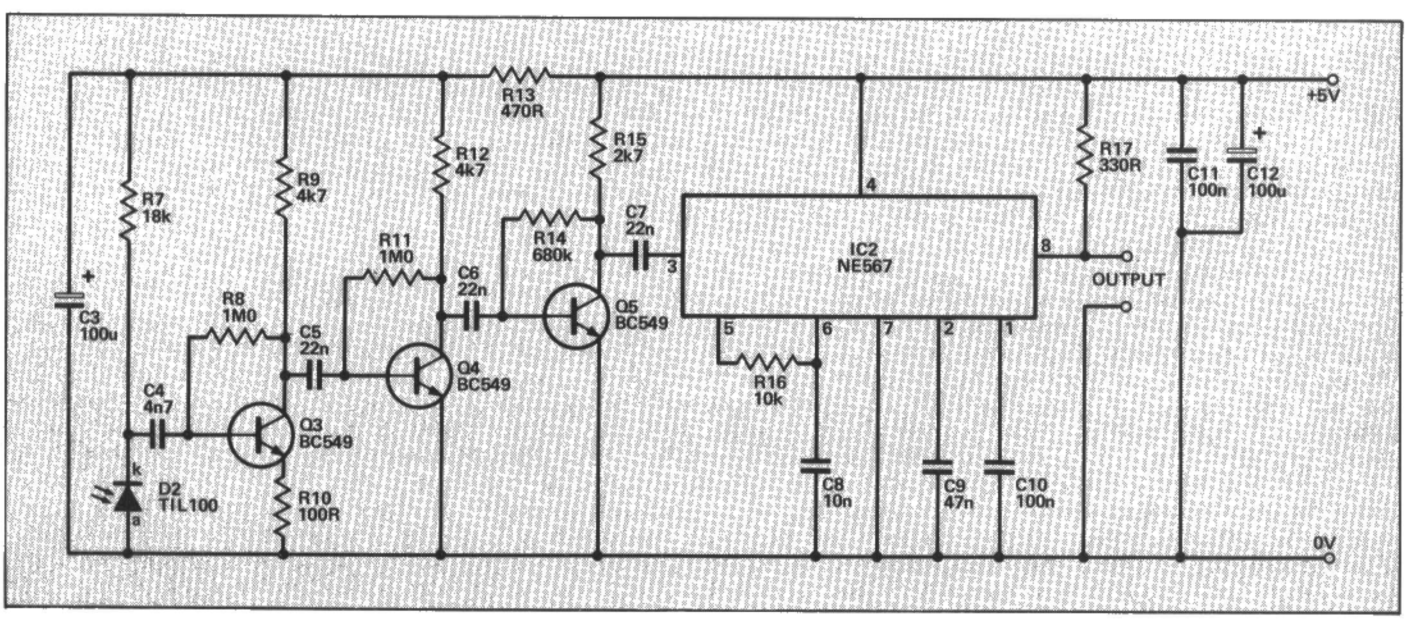
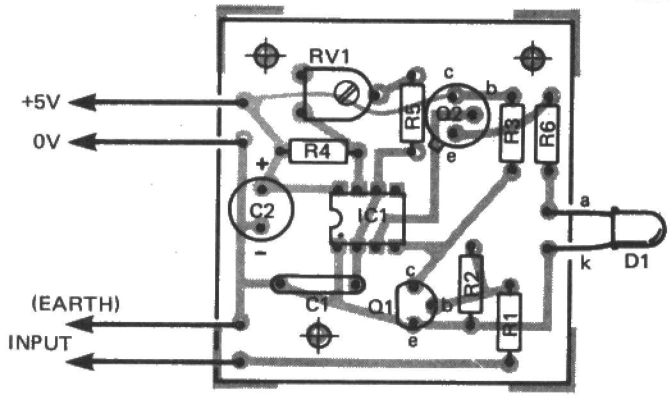
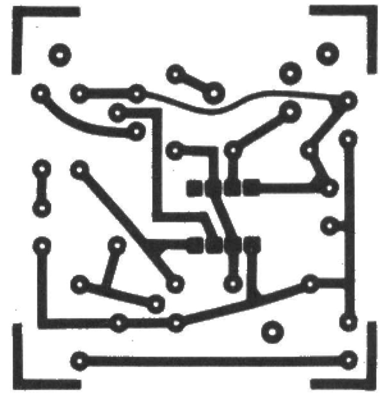
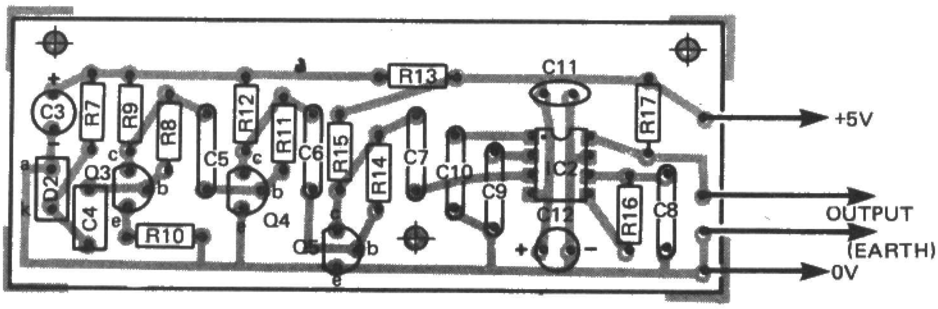


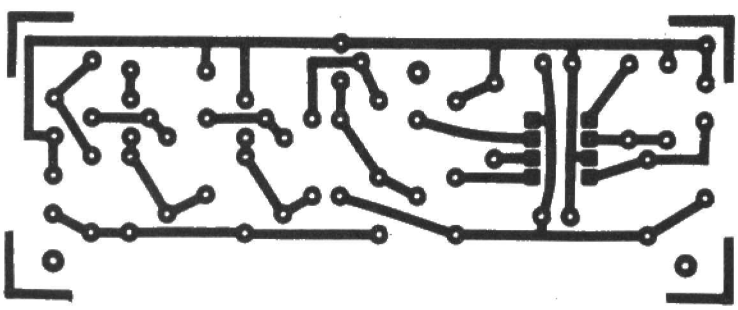
Figure 3. Receiver circuit diagram.

Figures 4 and 5.
Left, transmitter PCB and overlay.
Right, receiver PCB and overlay.









The only point to watch is that D1 is mounted with the sensitive surface (the one opposite the surface carrying the type number etc.) facing away from C4, TR3, etc. Of course, D2 can be mounted off-board if this would be more convenient, but due to the high sensitivity of the circuit it must be connected to the board via a piece of good quality audio screened cable.

When in use the infra-red output from D1 must be directed towards the sensitive surface of D2, but at short distances even with D1 aimed well off target the system will probably operate well. However, when used near the maximum range it is essential for the unit to be optically aligned properly, and for VR1 to be adjusted correctly. When initially testing the system, place the two units quite close together with D1 aimed at D2, and leave the non-earth input of the transmitter unconnected (so that the oscillator is enabled). If the output voltage of the receiver is monitored, it will almost certainly be about 5 volts, but by carefully adjusting VR1 it should be possible to lock the receiver onto the transmitter's output signal, and the output of the receiver high and low should then result in the receiver's output always switching to the same state, but if the circuit proves to be unreliable slight readjustment of VR1 should cure this. It is then advisable to move the transmitter and receiver two or three metres apart and then repeat this procedure to ensure that VR1 is set for optimum performance.

Although only one diode is used in the prototype transmitter, increased range and reduced directivity can be obtained by using three or four diodes. TR2 is capable of driving this number of diodes without difficulty, but these should each have their own 22R series resistor, and they should be connected in parallel. Bear in mind that the current consumption will increase by about 50 milliamps per additional diode.

Obviously the way in which the system is used will depend upon the particular application you have in mind. In computing applications the system is most likely to be used with RS232C style serial data, and a UART would then be used to convert parallel data into the desired serial format for transmission. Although not designed for use with signals at RS232C levels (nominally −12 and +12 volts), the transmitter seems to work perfectly well with standard RS232C signals and it can therefore be driven from the serial port of a computer. At the receiver another UART would be used to convert the serial data back into its original parallel form. Although not at proper RS232C levels, the output of the receiver will in fact directly drive the serial input of most computers properly, with the provision that a short connecting cable is used.

An important point to bear in mind is that computer equipment is very good at generating electrical noise, and if picked up by the receiver due to some form of stray coupling the useful range of the system could be greatly reduced. If necessary, the receiver board must be adequately screened and the supply to the receiver must be reasonably noise free.

Parts list (Transmitter)

Resistors (all ¼W 5%)	
R1,2	10k
R3	4k7
R4	1k
R5	1k5
R6	22R
Potentiometer	
VR1	10k 0.1W horizontal preset
Capacitors	
C1	10nF polyester
C2	470uF 10V radial elect
Semiconductors	
IC1	NE555
TR1	BC547
TR2	BFY51
D1	TIL38

Miscellaneous
Printed circuit board, 8 pin DIL socket, Veropins, wire, etc.

Parts list (Receiver)

Resistors (all ¼W 5%)	
R7	18k
R8,11	1M
R9,12	4k7
R10	100R
R13	470R
R14	680k
R15	2k7
R16	10k
R17	330R
Capacitors	
C3,12	100uF 10V radial elect
C4	4n7 carbonate
C5,6,7	22nF polyester
C8	10nF polyester
C9	47nF polyester
C10	100nF polyester
C11	100nF ceramic
Semiconductors	
TR3,4,5	BC549
IC2	NE567
D1	TIL100

Miscellaneous
Printed circuit board, 8 pin DIL IC socket, Veropins, wire, etc.

SIMU-LEDS

For those of you who like to keep the inner workings of your Beeb under observation, Mike Williams has written two short programs to show what those bits are doing at the user port.

Look inside a computer and it is hard to believe that anything is actually happening. As with most electronic devices the electrons go about their business and don't advertise their presence. Hence the array of meters, oscilloscopes and other tools that we use to show us what is going on. When experimenting with the user port for example, one of the first projects for many people is the construction of a bank of LEDs to the front panel of indicator lamps without which early computer hackers felt lost; the programs described in this article will allow you to do just that, and without any hardware to get in your way. They are of particular value if you are working with the User Port for control or monitoring purposes and would like to be able to see the current state of the various bits.

Simu-LEDs

The programs make use of the event routines provided by the BBC MOS. These were mentioned briefly in Mike James article on the BBC MOS *(E&CM December 1983)*. When certain events occur, such as a key being pressed, the operating system interrupts what it is doing and jumps via a vector to a set of routines which deal with the event. In the case of the key press for example, the key value may be stored in a buffer. That is why you can type into the computer even while it is running a program and it stores what you have typed in.

Certain events can be re-directed so that the user can make his own thing happen when the event occurs. In my programs I wanted eight simulated LEDs to appear on the screen showing the state of the output port. By using the screen refresh event which happens every fiftieth of a second, the 'LEDs' will in effect continuously display the port state.

Listing 1 gives the program to do this. It works in either Mode 1 or Mode 5. The memory location which is to be observed is chosen in line 240. Once the program is run, the 'LEDs' are visible at the top of the screen and they will continue to monitor the chosen memory location until Break is pressed. Changing to Mode 5 gives larger LEDs, albeit non-circular ones.

The code is assembled at &C00. It is fully relocatable and line 230 could DIM code%150, but I like it tucked out of the way. PROCinitialise first disables the video event just in case it is already running. Nasty things could happen in that case. After setting up some variables, the LED shapes are input with a pling*. If you want to use SimuLEDs in Modes other than 1 or 5 then you will have to modify the pling values as well as the screen addresses in lines 570 to 620.

The assembled program starts up setting the event vector at &220 to point to the code starting at line 530. The current Mode is tested (on OS1.2 the Mode is stored in &355), if it is neither 1 nor 5 the routine returns, otherwise the appropriate screen addresses are set up.

The memory location is then read and is shifted bit by bit to the left. Each time it is shifted, a bit drops into the carry. If it is a 1 then a nice red LED is poked onto the screen, otherwise an unlit LED is shown. The screen address is increased to give a line of LEDs and after 8 bits the routine ends.

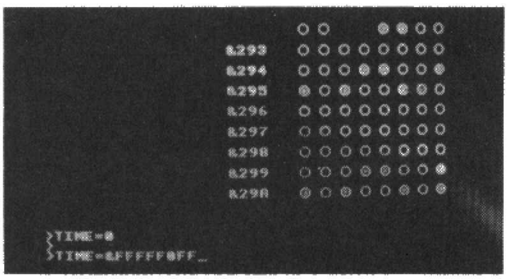
Despite the advice in the User's Guide (P.465) it only appears to be necessary to save the X-register. I have had no problems doing this.

**Non-hackers note: A pling is the '!' symbol used on the BBC as the word indirection operator, affecting four bytes.*

SIMU-LEDS 1

```
100REM  ************************
110REM  **      SIMU-LEDS     **
120REM  **         by         **
130REM  **    M.E.Williams    **
140REM  ************************
145:
150MODE1
160PROCinitialise
170PROCassemble
180CALLcode%
190END
200:
210DEF PROCinitialise
220*FX13,4
230code%=&C00 :      REM modify as necessary
240memloc=&FE64 :    REM  choose the byte to be observed
250PRINTTAB(18)"&";~memloc
260led=code%+&70
270noled=code%+&80
280screen=&80:       REM &80 and &81 only page 0 used by this routine.
300VDU28,0,31,39,1 : REM  Mode 1 text window.For Mode 5 - 0,31,19,1
310:
320REM Led on
330!led=&9E8F4733
340led!4=&33478F9E
350led!8=&971F2ECC
360led!&C=&CC2E1F97
370:
380REM Led off
390!noled=&88884433
400noled!4=&33448888
410noled!8=&111122CC
420noled!&C=&CC221111
430ENDPROC
440:
450DEF PROCassemble
460FOR PASS=0 TO 2 STEP 2
470P%=code%
480[OPT PASS
490LDA #(code%+18) MOD 256:STA &220 \ redirect Event vectors.
500LDA #(code%+18) DIV 256:STA &221
510LDA #14:LDX #4:JSR &FFF4 \ enable video sync event
520RTS
530TXA:PHA
540LDA &355              \current mode
550CMP #1:BEQ mode1
560CMP #5:BNE home
570LDA #&0:STA screen
580LDA #&58:STA screen+1
590BNE start
600.mode1
610LDA #&70:STA screen
620LDA #&31:STA screen+1
630.start
640LDA memloc:STA tempAstore
650LDX #8                 \number of bits in a byte
660.next_bit
670CLC
680LDA screen              \next led screen address
690ADC #32:STA screen
700LDA screen+1
710ADC #0:STA screen+1
720LDY #15                 \size of the led shapes
730ASL tempAstore
740BCC led_off
750.on LDA led,Y
760STA (screen),Y
770DEY
780BPL on
790DEX
800BNE next_bit
810BEQ home
820.led_off
830LDA noled,Y
840STA (screen),Y
850DEY
860BPL led_off
870DEX
880BNE next_bit
890.home PLA:TAX:RTS
895.tempAstore BRK
900]
910NEXT PASS
920ENDPROC
930REM start?1=memloc MOD 256
940REM start?2=memloc DIV 256
```

The memory location given in line 240 is that of one of the VIA timers, and I chose it so that you can see something going on when you first run the program. By changing it to &FE60 you can monitor inputs and outputs on the user port. In that case you *SAVE the assembled code and *RUN it as part of a program involving the user port. By setting the text window as shown, the LEDs can be isolated at the top of the screen. It is possible to change the address being examined by directly poke-ing it in. To do that you should first find the address of 'start'. (If the routine is at &C00 then start = &C31). Then:



?(start+1)=memloc MOD 256
?(start+2)=memloc DIV 256

The routine can be disabled at any time by *FX13,4 and can be re-enabled by *FX14,4.

As a quick check to ensure that the software is running correctly, change line 240 to memloc=&FE62 and RUN. Then:

FOR A% = 0 to 255: ?&FE62 = A%: FOR B% = 0 to 2000: NEXT B%: NEXT A%

The routine counts from 0 to 255 over the eight simulated LEDs, and includes a delay loop to allow the changing light sequence to be visible.

Simu-LEDs 2

The first Simu-LEDs program enables us to monitor one memory location. If you have ever seen Computer Concepts Disc Doctor chip pointing its MZAP function into various parts of the memory map you will have observed that some bytes are constantly changing. Simu-LEDs 2 allows you to monitor eight consecutive memory locations and reveal the changing bit patterns.

The program is similar to Simu-LEDs 1; it had to be modified to allow 8 successive locations to be loaded into the accumulator and then examined. The copious assembler elements should make its operation clear.

Once assembled the screen should show eight rows of 'LEDs' starting at address &293. These should be changing, especially &29A. They are in fact used by the TIML routine. Try the following:

Press <escape> The LEDs should still be working

Enter – TIME 0 LEDs &276 to &29A should clear

Now you can observe the 'clock' counting the centiseconds.

Enter – TIME=&FFFFFOFF sets the LEDs after a while to &296 increments.

Another 'clock' starts at &295.

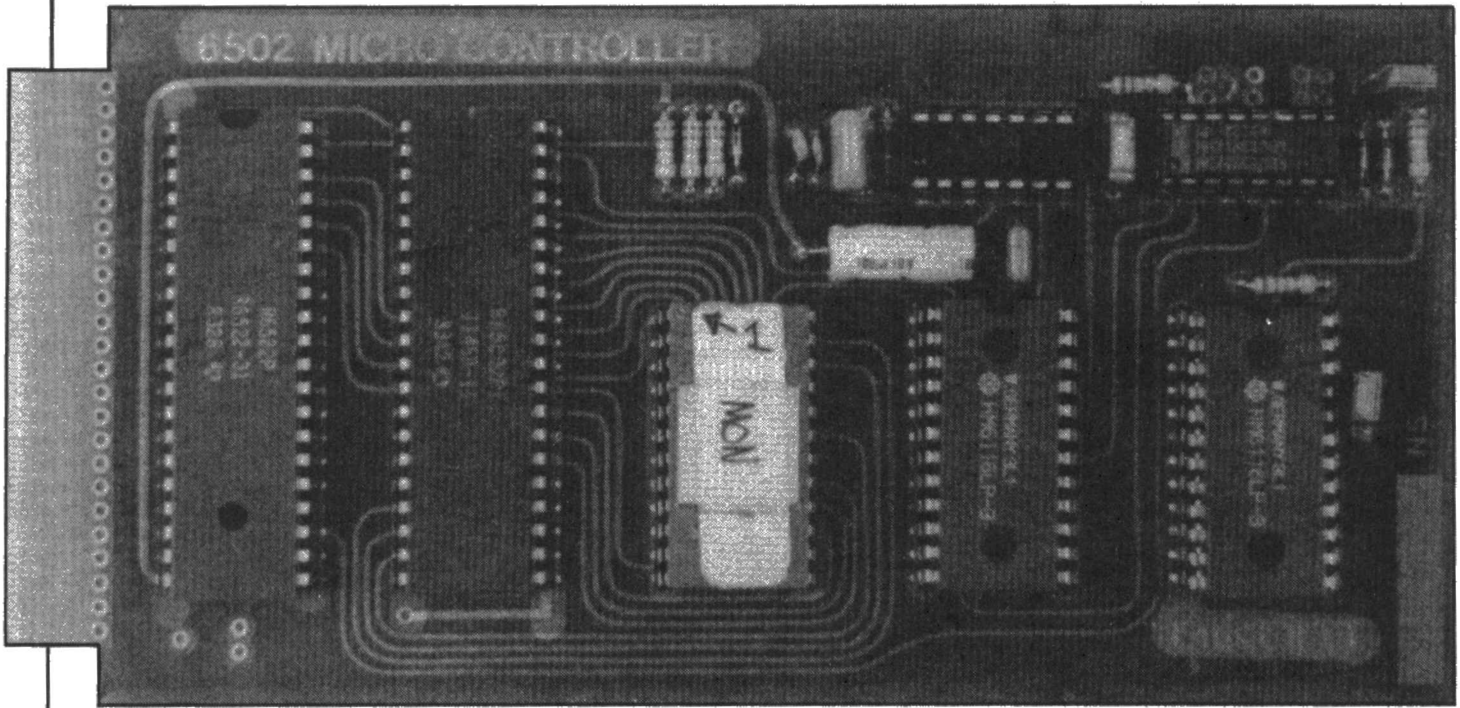
Other fruitful areas for investigation are:

&1D0 Part of the stack. Try scrolling here
&FE40 The system VIA
&FE60 The User Port

The program could be modified to allow scrolling through memory, but I leave it for you to implement. Happy hunting! ■

SIMU-LEDS 2

```
100REM  **************************
110REM  **       SIMU-LEDS2     **
120REM  **          by          **
130REM  **     M.E.Williams     **
140REM  **************************
150:
160MODE1
170ON ERROR GOTO 1350
180PROCinitialise
190PROCassemble
200PROCaddresses
210CALL code%
220:
230REPEAT
240PRINTTAB(2,20)"              "
250INPUTTAB(2,20)"Memloc &"memloc$
260IF memloc$="" THEN 250
270memloc=EVAL("&"+memloc$)
280IF memloc<&FF THEN PRINT"Not zero page please.":GOTO 240
290CLS
300PROCaddresses
310UNTIL FALSE
320END
330:
340:
350DEF PROCinitialise
360REM disable video synch event.
370*FX13,4
380code%=&C00 : REM fully relocateaole
390:
400led=code%+&80
410noled=code%+&90
420memloc=&0293 : REM  something to look at.
430screen=&80: REM routine uses &80 and &81 from 0 page.
440:
450REM Led on
460!led=&9EBF4733
470led!4=&33478F9E
480led!8=&971F2ECC
490led!&C=&CC2E1F97
500:
510REM Led off
520!noled=&88884433
530noled!4=&33448888
540noled!8=&111122CC
550noled!&C=&CC221111
560ENDPROC
570:
580DEF PROCassemble
590FOR PASS=0 TO 2 STEP 2
600P%=code%
610[OPT PASS
620LDA #(code%+18) MOD 256:STA &220  \ change event vectors
630LDA #(code%+18) DIV 256:STA &221
640LDA #14:LDX #4:JSR &FFF4          \ enable video sync event
650RTS
660TXA:PHA              \save X register
670LDA &355             \current mode
680CMP #1
690BNE home             \ enter routine only if Mode 1
700LDA #&70
710STA screen           \set screen position for bottom led
720LDA #&59
730STA screen+1
740.start
750LDY n_of_bytes       \number of locations to examine
760LDA memloc,Y
770STA tempAstore
780LDX #8               \number of bits in a byte
790.next_bit
800CLC
810LDA screen            \next led screen address
820ADC #32
830STA screen            \move 2 spaces right
840LDA screen+1
850ADC #0
860STA screen+1
870LDY #15               \size of the led shapes
880ASL tempAstore
890BCC led_off
900.on
910LDA led,Y
920STA (screen),Y
930DEY
940BPL on
950DEX
960BNE next_bit
970BEQ next_memloc
980.led_off
990LDA noled,Y
1000STA (screen),Y
1010DEY
1020BPL led_off
1030DEX
1040BNE next_bit
1050.next_memloc
1060DEC n_of_bytes
1070BEQ home          \ 8 memory locations done
1080LDA #&70
1090STA screen
1100SEC               \ move up 2 lines
1110LDA screen+1
1120SBC #6
1130STA screen+1
1140JMP start
1150.home
1160LDA #8
1170STA n_of_bytes       \ready for next time
1180PLA:TAX
1190RTS
1200.n_of_bytes BRK
1210.tempAstore BRK
1220]
1230NEXT PASS
1240?n_of_bytes=8
1250ENDPROC
1260:
1270DEF PROCaddresses
1280?(start+4)=memloc MOD 256:?(start+5)=memloc DIV 256
1290FOR J%=0 TO 7
1300PRINTTAB(18,J%*2+2);"&";~(memloc+J%)
1310NEXT
1320ENDPROC
1330:
1340REM if escape, set text window below leds
1350IF ERR=17 THEN VDU28,0,31,39,20:CLS:END
1360REPORT
1370PRINT" in line ";ERL
1380END
```



6502 MICROCONTROLLER

The Nikam 65C02 controller for the BBC puts an end to cracking control problems with a sledgehammer. Vincent Fojut puts the board to the test.

The current low price and sophistication of microprocessors make them the most cost-effective (and powerful) candidates for many dedicated control applications, with robotics being a prime example. However the development of such projects can often present a dilemma to the average hobbyist. On the one hand, the compact hardware of a small, single-board microcomputer may suit the application admirably, but software support is often very crude. Conversely, personal micros like the BBC computer have excellent software development facilities, but are not necessarily appropriate from a hardware point of view. (Can you honestly imagine a micromouse carrying a Beeb on its shoulders?).

A neat way of resolving the conflict is offered with a new microcontroller board, from Nikam Electronics. Not only is the board small and light enough for most demands, but it also provides a "software UART", which allows program development to be carried out on the BBC micro, and downloaded into the microcontroller via the RS423 port.

Package details

The microcontroller circuit board is certainly compact, measuring a mere 6 x 3 inches. The board is populated with a 65C02 microprocessor, up to 4K of CMOS Ram (6116's), a 2K monitor in a 2716 Eprom, and a 6522 VIA providing I/O and timers. To allow connection with the BBC micro for software development, the board plugs via a 24-way edge-connector into a small motherboard. This houses a voltage converter and buffers for RS423 I/O levels, plus a reset switch, and Mulex connectors for interfacing the VIA ports. An RS423 connector and lead are also provided. The required +5 volt supply can be taken from the Beeb's PSU, or any other suitable source. A manual completes the package, containing various notes, circuit diagrams, memory map, and three BBC program listings.

The first of these is a short "terminal" program (see **Listing 1**), needed to interact with the Nikam microcontroller via the serial port. The second details a "fast loader", which is used to download a continuous block of assembled machine code from the Beeb into the microcontroller's user Ram, at &4000-&47FF. This can be made to simulate Eprom during "in situ" program testing. Finally, a short demonstration program is supplied, which, with the aid of a small amount of hardware, allows the user to familiarise himself with the system, and ensure satisfactory operation.

Monitor commands

The monitor supplied comes with a range of 5 simple commands (see **Table 1**) designed to provide a minimum level of support for the BBC RS423 interface. The following facilities are covered:- memory dump, memory load, fast memory load, execute ("Go"), and single-step.

So, what are the stages involved in developing a control application? Firstly, using the BBC microcomputer's built-in 6502 assembler, programs can be written, and, to some extent, tested and debugged on the BBC micro itself – for example, using the user-port to interact with the external devices. The time eventually comes, however, when the program needs to be tested with the very hardware on which it will ultimately run. Often, in this case, the only option available to the

"The Nikam board is small, light and provides a software UART".

hobbyist is to burn the program onto eprom, plug it into the dedicated device, and hope for the best. When the inevitable bug appears, the eprom is erased, reprogrammed, and the cycle repeated. If several bugs are present (and they are rarely solitary creatures!), the whole process can be very frustrating, not to mention time-consuming.

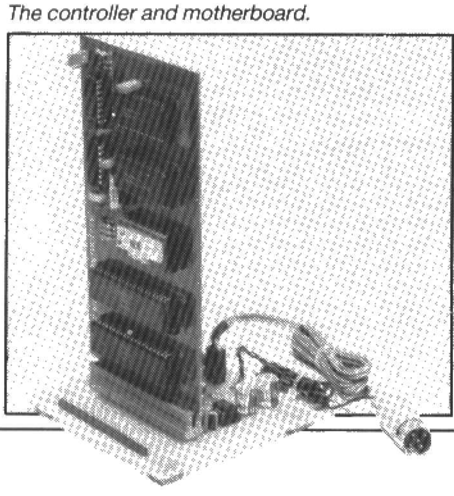
Fortunately, the Nikam microcontroller is far more convenient to use. Once a program is assembled on the Beeb, it can be downloaded into the microcontroller's RAM (using the "Fast Load" command), then made to behave exactly like an eprom. This is achieved by changing two jumper links on the board.

The first of these links determines from which address the reset vector is retrieved. In its default position, this is normally at &FFFC and &FFFD (in the monitor eprom). However, if user RAM is at its usual address of &4000 to &47FF, then once the link is moved, the reset vector will now be held in &47FC and &47FD. These should contain the low and high order start address of the user-written control program. If the reset switch is then pressed, the program automatically runs from the start address held in the new reset vector.

The second jumper link puts the user RAM in "write-protected" mode, thereby preventing the risk of a wayward program overwriting itself! In this way, one can emulate eprom as closely as possible, yet still modify code with relative ease. If bugs are discovered at this stage of development, it is far easier to modify the assembler source code, and reload into the microcontroller's RAM than to erase and reprogram an eprom. Once satisfied with your code, you can burn the control program onto eprom, insert it in place of user-RAM, and remove the monitor, which is no longer required. You are safe in the knowledge that the control program has been tested on the very same hardware, and in the same locations. Experienced programmers could even relocate the code to occupy the monitor's memory space. The control program then replaces the original monitor, leaving a full 4K of RAM space available.

The Nikam microcontroller must be one of the first commercial products to use the new 65C02 microprocessor. This is an enhanced CMOS version of the popular 6502 chip, which not only offers extremely low power consumption, but also has a range of new machine code instructions (see **Table 2**). Furthermore, some existing instructions have been given extra addressing modes. Of course, these new instructions are not explicitly supported by the BBC assembler, but suitable procedures can be written to get round this. Indeed, many of the new instructions (such as bit test, set & reset), are particularly well suited to control program requirements.

The controller and motherboard.



Currently the board is supplied with standard versions of the 2716 eprom and 6522 VIA, but if CMOS versions of these devices are used, then total board consumption is in the region of 25ma. This makes portable, battery-powered equipment a perfectly feasible (and attractive!) proposition.

Drawbacks

Putting the monitor program through its paces showed up a number of shortcomings. Most curious of these is the single-step command, which displays the values of all internal registers – except the stack pointer! Also the value displayed for the break flag appears inconsistent. In addition, the monitor does not handle the delete key correctly, so you really need to get things right first time, or re-enter the whole line.

Being curious to see how the monitor was coded, I would have liked a command to "upload" data from the microcontroller into the Beeb. In this way, even the monitor itself could be disassembled. As an alternative, I tried creating a *SPOOL file of a hex dump of the monitor's memory area, using the memory dump command. However, the command would not work reliably whenever *SPOOL was used, possibly due to some form of synchronisation problem. In all, the monitor is a weak link, and does not reflect the quality of the rest of the system, which otherwise appears good. Nonetheless, the monitor does meet its limited objectives, and is certainly workable, providing you are willing to live with the constraints outlined above.

Assessment

The ease with which programs can be modified during testing is the system's great strength. I did find the jumper links a little fiddly to use. A hardware switch would make the operation easier, but possibly at the cost of increasing board size. It's also easy to forget to change links back to their original positions, in order to be able to modify code (you can't change write-protected RAM!).

The Nikam microcontroller merits consideration by the robotics enthusiast, or any other dedicated system builder. An enhancement to the monitor software would undoubtedly produce a more friendly product. Nevertheless, the relative ease of program development, together with its size, lightness, and low-power potential, could win the microcontroller a lot of friends. Finally, the availability of bare boards should make it particularly interesting to those on a limited budget.

The 65C02 Microcontroller is available from *Nikam Electronics Ltd., 25 Suffolk Drive, Lacey Green, Wilmslow, Cheshire SK9 4DE.* Prices are as follows:-

Assembled & Tested Microcontroller, with monitor, 4K Ram, & manual	**£84.98**
Bare board	**£16.99**
Monitor Eprom	**£14.99**
Motherboard	**£18.99**
Bare Motherboard	**£4.99**
Technical Manual	**£2.50**

LISTING 1. Simple BBC/Microcontroller communication program.

```
10 REM TERMINAL
20 MODE 7
30 REM TRANSMIT 2400 BAUD
40 *FX8,5
50 REM RECEIVE 2400 BAUD
60 *FX7,5
70 REM TAB TO ESCAPE
80 *FX220,09
90 st%=&FE08:REM STATUS
   REGISTER
100 trx%=&FE09: REM TRANSMIT/
    RECEIVE
110 IF (?st% AND 1)=1 PRINT
    CHR$?trx%)
120 a$=INKEY$(0):IF a$<>""
    ?trx%=ASCa$
130 GOTO 110
```

TABLE 1. Monitor commands (all parameters in hex).

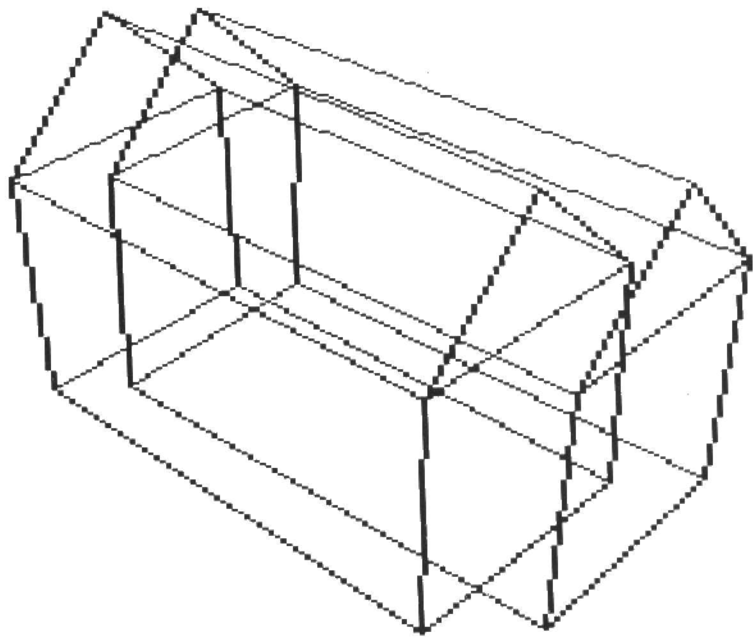
Symbol	Funtion
.	Memory dump eg 4000.10 displays 16 bytes from address &4000 as 2 lines of 8 bytes
:	Memory load eg 4000:A9 01 stores two bytes, &A9 and &01, starting from address &4000
#	Fast memory load – unloads block of memory under program control. Sample program provided
G	Execute program ("Go") eg 4000G runs program starting at address &4000
S	Single-step eg 4000S steps through program starting at &4000, one instruction at a time, displaying registers & flags.

TABLE 2. New 65C02 instructions

Mnemonic	Operation
BBR	Branch on bit reset
BBS	Branch on bit set
BRA	Branch always
PHX	Push X register onto stack
PHY	Push Y register onto stack
PLX	Pull X register from stack
PLY	Pull Y register from stack
RMB	Reset memory bit
SMB	Set memory bit
STZ	Store zero
TRB	Test and reset bits
TSB	Test and set bits

3D stereo vision

Mike James takes the quest for realism one stage further with computer generated 3D stereo vision, hidden lines, and shadowing.



There are many applications where the sort of 3D computer graphics that have been described so far in this series are more than good enough. However, the quest for realism in computer graphics is never ending. The simplest goal is to create life-like images that can be used in simulators or as part of a computer-aided design process. At the other extreme the aim is to produce 'super real' images that are part of a new type of art or extend the possibilities of TV and cinema.

In some ways the task of producing the latter type of computer graphics is often easier from a technical point of view because of the tolerance of the human viewer. For example, the feature film TRON which contained a great deal of computer graphics was only possible because the effects didn't attempt to mimic reality – the graphics created a fantasy landscape that was very easy to accept. Compare this, for example, with the difficulty of creating a simulation of the sea breaking against a beach – even a powerful computer like the Cray 1 takes minutes to generate such images.

In the realm of the personal computer the current state of technology does limit the degree of realism that can be achieved in 3D graphics. Some special hardware for realistic 3D graphics is now being produced at prices that suggest that in the not-too-distant future, personal graphics computers will be available. However, for the moment, there is nothing much we can do apart from wait or spend a great deal of money! It is not difficult to think up fairly simple hardware projects that will produce the required display flexibility and resolution and so the only real problem is processing power. If you want to create realistic scenes then you have a great many calculations – so many that even the fastest of today's computers take minutes per scene. They are still too slow, for example, to make realistic interactive 3D graphics possible. The new generation of computer games are successfully producing very realistic interactive graphics but this is achieved with the aid of a laser video disc which is used to store complete frames of the action. In the future such fast, high capacity storage devices may make it possible for personal computers to generate realistic interactive graphics but for the time being we are restricted to cartoon-like images.

While it is possible to produce 3D graphics that look very good using a personal computer, it is not possible to approach the realism of even a black and white photo unless you use a TV camera or 'computer-assisted painting'. There are special types of images that can be created quickly by using processing tricks and short cuts. However, if you want to write a program that will take a descripton of a scene and then produce an image that represents what it looks like from a particular viewpoint, then the 3D viewer is about as far as you can go using a personal computer and BASIC. Even if you change to assembler there are still processing problems that limit what you can do. The first part of this article looks at some of the most difficult topics in graphics – hidden line and hidden surface elimination, shading, shadowing and textures. The final part returns to the realm of what can be done with a micro in a rather surprising way – scenes with realistic shading etc, it is remarkably easy to produce images that appear to have true 3D depth and 'float' either in front or behind the TV screen!

"... the 3D viewer is about as far as you can go using a personal computer and BASIC".

Hidden line and surface removal

The 'wire frame' images produced by the 3D viewer **(Listing 1)** serve to represent objects fairly well once you get used to interpreting them. Unfortunately, there are times when being able to see through an object produces ambiguities. For example, there is a well known visual illusion (known as the Necker cube) that results from the two possible interpretations of the wire frame cube **(Figure 1a)**. This ambiguity can be reduced by exaggerating the perspective used in drawing the cube **(Figure 1b)** but sometimes even this fails. The reason for the Necker cube illusion is simply that it is difficult to tell which face of the cube is in front of the other and the most direct method or resolving this ambiguity is to change from 'see through faces' to solid faces. Drawing the cube with solid faces amounts to not drawing those lines that are hidden by the faces closest to the observer **(Figure 1c)**. Removing hidden lines seems a very easy task for a human but for a computer it is very difficult. In everyday life, hidden line removal occurs automatically as a result of the way in which we see things by reflected light. That is, if one object is in front of another, it will block the light rays from the object it is in front of and stop them from reaching the eye of the observer. However, when it comes to a computer's representa-

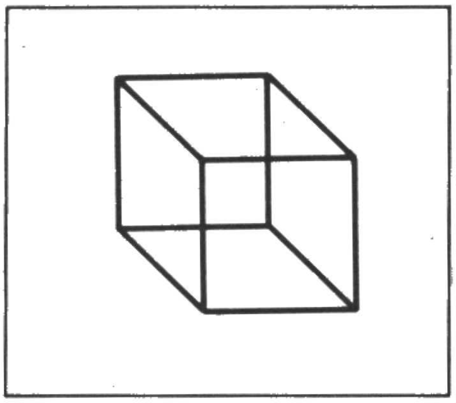


Figure 1a. The Necker cube illusion. Which is the front face?

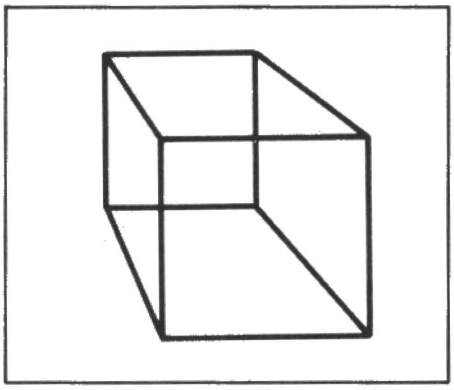


Figure 1b. Adding perspective helps to identify the front face.

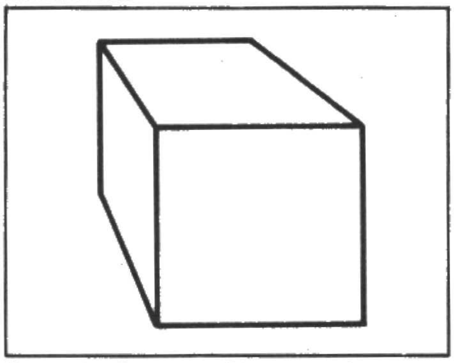


Figure 1c. Removing 'hidden lines' makes it even more obvious.

tion of the same objects, there is no natural process that will result in the closer matter obscuring part of the other object – you have no choice but to work out what is visible.

The main problem with hidden line removal as a calculation is that it needs information about the positions of all the planes or faces that make up the object. With a simple wire frame representation we have been able to make do with the co-ordinates of points and information about the lines that join them, but, for hidden line removal, it is necessary to consider how the lines group together to form faces. This increases the amount of information that we have to store to represent an object as well as the number of pixels that are involved in drawing it. When a simple wire frame representation of an object is drawn, only the pixels that form the lines between the points are involved but, when hidden line removal is used, then every pixel within a face is involved. For example, one of the simplest hidden line removal algorithms is the "painter's algorithm". This is based on the observation that if each face is drawn as a solid face (that is, by changing the colour of each pixel within it) then the faces that are drawn later will automatically obscure those drawn earlier. If it is possible to arrange things so that the order in which they are drawn corresponds to their depth, then closer faces which are drawn later will obscure the earlier, more distant faces and the result is automatic hidden line removal. The term "painter's algorithm" comes from the way a painter will often paint the foreground over the top of the background. You should be able to see that this method requires every pixel within a face to be altered and not just the lines that form the boundary of it. This fact alone means that drawing solid representations of objects is a slow process but also, in addition, time is required for working out the order that the faces should be drawn in. All hidden line methods involve a stage of sorting faces into depth order – or 'geometric sorting' as it has come to be known – and this is where most of the current research work into hidden line removal is concentrated at the moment. Geometric sorting is very difficult because each face in a scene may occupy a range of depths and it is difficult to decide which should be drawn first. Indeed it may not even be possible to sort faces into order (see **Figure 2**) and then the only real choice is to divide the faces down into smaller pieces and try again.

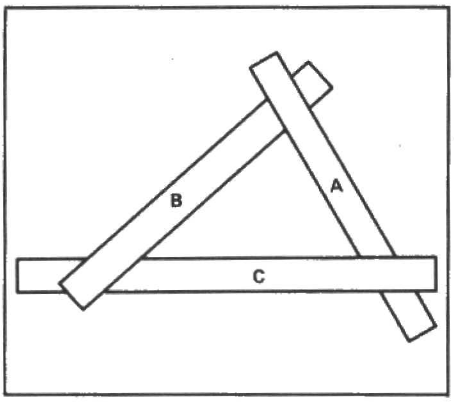


Figure 2. 3 faces that are impossible to sort out into depth order. Which should be drawn first?

The painter algorithm is a good method of hidden line removal and it is often used in conjunction with sophisticated geometrical sorting algorithms. It can be particularly effective when the scene being generated may be easily sorted by depth. For example, if you are drawing a mountainous landscape, all you have to do to perform hidden line removal is to draw the most distant mountain ranges first! However, even if the geometrical sorting takes little or no time, the problems of colour filling areas is something that micros are fairly slow at. For example, the Dragon's PAINT command is easy to use but takes around a second to fill any medium sized area, the BBC Micro and Electron's fill commands are faster but still make the painter algorithm a long wait! As already mentioned, even if you use a different method, you still cannot avoid examining most of the pixels that make up a face. For example, the depth buffer method of hidden line removal examines each pixel on the screen in turn and assigns it the colour of the face closest to the observer that passes through it. This sounds easy but, once again, the difficulty lies in discovering which face is closest to the observer at any given point. In practice this means examining each face in the scene and discovering which ones contain the pixel in question. Then, for each face that contains the pixel, the depth at that point is worked out. The pixel then takes the colour of the closest face; this algorithm is easy but it involves examining every pixel on the screen and every face in the image a great many times.

"The ultimate in realism is obtained using shadowing".

You should be beginning to see that hidden line removal is not a quick job! However there are one or two very simple forms that work for certain types of image. For example, if you are only interested in convex shapes (ie shapes that have no indentations) then you can use the fact that you can always see the 'outside' view of a face. Put simply, this algorithm decides which side of a face is visible by using the order of its corners. For example, if you number the corners of a face in a clockwise direction when looking at it from the front or outside, then the corners will appear to be numbered anti-clockwise if you look at it from the other side. Using this, or a similar convention, a graphics program can examine the order of the corners of each face that it is about to draw and only draw those that are viewed from the outside. This hidden line elimination method is fast and efficient but it only works for convex objects – and most interesting objects are not convex.

Shadow and shadowing

The ultimate in realism is achieved when the faces of an object are shaded or coloured in a way that corresponds to how they would look if illuminated by a given source of light. Surprisingly the calculations necessary to determine the brightness of a point are not at all complicated and, if used with a hidden line removal algorithm, add little to the time taken to produce an image. For personal computer use the real problem is finding enough colours or grey levels to produce a realistic image. In some senses this is simply a hardware problem in that the display generators only work with 4 or 8 colours but it is really a reflection of the fact that today's micros would be hard pushed to manipulate displays with a greater colour resolution. Even the BBC Micro at its highest colour resolution only manages eight colours on a 160 by 256 pixel screen.

Commercial graphics systems work with a minimum of 256 grey levels or colours and 256 by 256 pixels is usually considered the minimum spatial resolution!

The shading of an object is obviously dependent on the type of illumination that it is subject to. In general it is useful to distinguish two types of illumination – diffuse and point source. Diffuse illumination is characterised by having a constant intensity independent of direction. In this case the amount of light that reaches the eye of the observer is given by:

Ed=RI

where R is a constant that governs the reflectivity of the surface and I is the light intensity. The use of only diffuse illumination produces very unnatural looking scenes and it is necessary to add a point source of light to improve things.

There are two ways that a point source of light can be reflected from a surface – diffuse reflection and specular reflection. Diffuse reflection corresponds to 'dull' surfaces and specular reflection corresponds to 'shiny' or mirror-like surfaces. Taking both types of reflection into account, the energy that reaches the eye of the observer is given by:

(R COS(i) + W(i)(COS)(s))^N) IP

where i and s are angles as shown in **Figure 3,** R is the reflection co-efficient, W(i) is a specular reflection function which varies according to material and angle of incidence and finally N controls how shiny the surface is. A value of N around 10 will produce silvery surface effects and a value around 1 will produce matt surfaces. You can also include transparency effects just as easily but the two equations given above are sufficient for many applications. For example, you can model the shading and reflections produced by the light from a window by representing it as a collection of point sources positioned in the shape of a window. Such calculations are easy in principle but time consuming in practice. Another apparently simple task is shadowing. It is obvious that objects illuminated by a point source will cast shadows but working out where they are and how they affect the illumination of other objects is very difficult. It is, in fact, a problem that is very similar to the hidden line problem described earlier. In this case the object is to discover which light sources are hidden by other parts of the image. Once again shadowing algorithms are much easier to apply as part of a hidden line removal program.

Ray tracing

After describing a number of 'traditional' algorithms for improving the realism of 3D images it is worth mentioning one new approach that at first seems to be a highly inefficient method – ray tracing. Ray tracing calculates the path of a ray of light through the scene to the observer's eye and so arrives at a value for its intensity. Thus the scene is built up in a way that copies nature. The only reason that this method has any hope of being possible in a reasonable amount of time is that the new super computers such as the Cray 1 can trace a number of rays in parallel and so speed things up by raw number crunching power. Ray tracing is hardly likely to appear as a method on personal computers until they pack the punch of a Cray 1 . . . any guess at when this will happen is almost certain to be wrong!

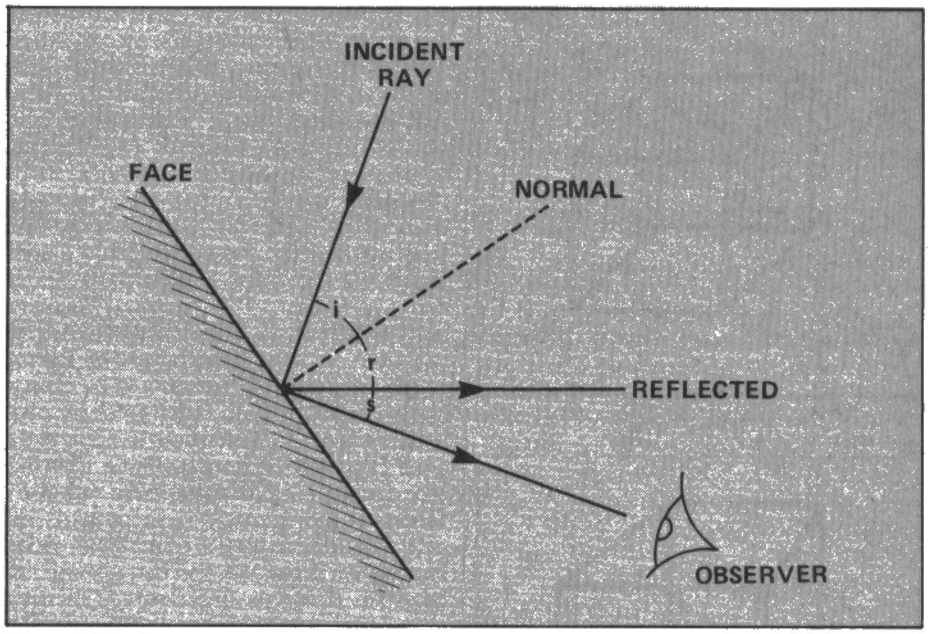


Figure 3. Shadowing.

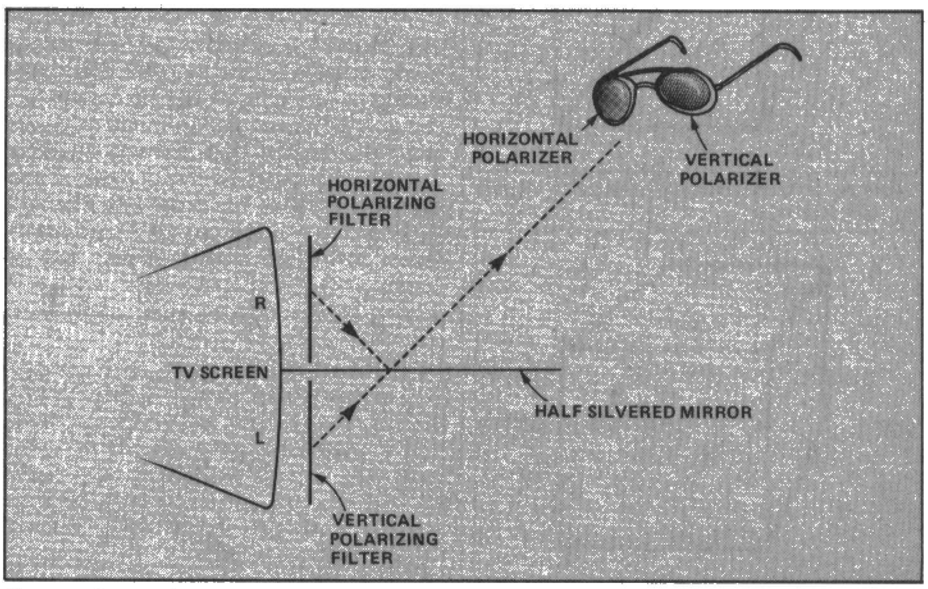


Figure 4. Stereo viewer using polarizing filters.

Stereo vision

Even if we stay with the simple wire frame images of objects produced by the 3D viewer (opposite), there is one method that can be used to improve the realism that is easy to implement and which only slightly increases the time taken. Normal human vision is in stereo, that is, not only do we see objects as single perspective views, but we fuse the perspective views produced by both eyes into a single 'solid' object. To produce the illusion of a computer-generated solid object all that we have to do is to display two perspective images – one corresponding to what would be seen by the left eye and one corresponding to the right eye, making sure that each eye sees only its particular image. This method of producing 3D drawings and even film and television images has been used for many years.

Image separation was traditionally solved by the use of red and green spectacles, and by drawing one of the images in red and the other in green. Thus one eye can only see the red image and one eye can only see the green image and the right and left view can easily be 'delivered' to the correct eye. For computer use this red/green system is particularly appropriate because most machines can generate high resolution images in red and green on a standard colour TV. However if you want to produce high quality 3D stereo images it is worth constructing a split screen system using polarising filters – see **Figure 4.** Polarising filters are much better at keeping the two images separate

and they don't introduce false colours.

For the purposes of demonstration and experimentation, however, it is easier to use the red/green colour system. For this you will need a pair of spectacles with a red filter in the left eye and a green filter in the right eye which you can easily make using coloured celluloid. The modifications to the 3D view program are simple enough. First the line drawing routine (subroutine 4000) has to be modified so that it doesn't clear the screen and so that the foreground colour can be set to either red or green. Then the perspective projection routine has to be modified so that it can produce two projections corresponding to what would be seen by the right and left eye. This can be achieved by shifting the centre of projection horizontally by an amount that represents the normal separation of the eyes. For the BBC micro version of the 3D viewer these changes amount to:

```
  20 MODE 2
  66 CLS
  70 DE=75:GOSUB 3000
  80 GCOL 0,2:GOSUB 4000
  90 DE=-75:GOSUB 3000
 100 GCOL 0,1:GOSUB 4000
 110 GOTO 35
1520 VDU 19,0,0,0,0,0
1530 VDU 19,1,1,0,0,0
1540 VDU 19,2,2,0,0,0
1550 RETURN
3010 A(I)+-X(I)*ZC+Z(I)*(XC+DE)
4000 REM DRAW
```

The only modification that needs any explanation is the introduction of the variable DE into line 3010. This is a horizontal displacement that is applied to the centre of projection before the perspective transformation is performed. Essentially it produces the displacement corresponding to the difference between the right and left eye. If you make DE smaller then the stereo effect will be less, if you make it bigger then the stereo effect will increase but it might prove difficult for the eyes to fuse the two images and so the effect might fail completely!

If you make these modifications to the 3D viewer and then look at the resulting images on a well-adjusted TV or colour monitor you will find that the 'house shape' now produces the illusion of depth. If at first you don't see a single image then allow time for your eyes to adapt to wearing red and green spectacles, darken the room and try to improve the contrast between the two colours. You will find that on most TV sets the difference between red and green is such that you can see the green image even through the red side of the spectacles and this tends to produce a second 'ghost' image within the stereo illusion, there is little that can be done about this apart from trying to match the filters to the colours more accurately or changing over to the polaroid system shown in **Figure 4.** The 3D viewer program allows you to move closer and further away from the house shape (by pressing the C and F keys) and to view it from 'behind'. If you look at the house using positive distances you will find that the stereo illusion is behind the TV screen. If you look at it when using negative distances then the house appears to float in front of the screen. Using negative distances produces images that are, at first, more difficult to fuse into a single image, but once you get used to it the effect is more pronounced than for images behind the screen.

The additions to the 3D viewer for producing stereo images are so slight that it is clearly pssible to use such illusions for real applications. The overall success and acceptability of such programs depends very much on the ease of viewing the illusion and if you are contemplating a real application my advice is to build the split screen polarising viewer.

Conclusion

Just what personal computers are capable of in the field of 3D graphics is still being explored. To a great extent I hope that some reader will prove my pessimism about the practicality of a general hidden line, shading and even shadowing algorithm misplaced by producing such a program! In the meantime there is a great deal of fun to be had with stereo views and constructing the special hardware to look at them.

LISTING 1. 3D viewer.

To our horror (no exageration here) we discovered on receiving copines of the June issue of *E&CM* that the wrong listing had crept, crawled or bribed its way into Mike James' Micrographics article. The correct listing, (referred to as Listing 1 in this month's article) which gives a 3D animated wire graphic representation, is printed below. The program is written in BBC BASIC, but none of the special features of the BBC have been used, and it can be easily adapted for any machine.

```
  10 REM 3-D VIEWER
  20 MODE 0
  30 GOSUB 1400
  35 GOSUB 1000
  40 GOSUB 2000
  50 GOSUB 5000
  60 GOSUB 6000
  65 GOSUB 7000
  70 GOSUB 3000
  80 GOSUB 4000
  90 GOTO 35
1000 RESTORE
1010 DATA 0,0,0
1020 DATA 0,0,2
1030 DATA 0,1,0
1040 DATA 0,1,2
1050 DATA 1,0,0
1060 DATA 1,0,2
1070 DATA 1,1,0
1080 DATA 1,1,2
1081 DATA .5,1.5,2
1082 DATA .5,1.5,0
1100 DATA 0,4
1110 DATA 4,6
1120 DATA 6,2
1130 DATA 2,0
1140 DATA 1,5
1150 DATA 5,7
1160 DATA 7,3
1170 DATA 3,1
1180 DATA 0,1
1190 DATA 4,5
1200 DATA 6,7
1210 DATA 2,3
1220 DATA 2,9
1221 DATA 3,8
1222 DATA 7,8
1223 DATA 6,9
1224 DATA 9,8
1300 P=10
1310 L=17
1320 FOR I=0 TO P-1
1330 READ X(I),Y(I),Z(I)
1335 X(I)=X(I)*200
1336 Y(I)=Y(I)*200
1337 Z(I)=Z(I)*200
1340 NEXT I
1345 X(P)=0:Y(P)=1:Z(P)=0:P=P+1
1350 FOR I=0 TO L-1
1360 READ S(I),F(I)
1370 NEXT I
1380 RETURN
1400 TH=PI/4
1410 PH=PI/180*35
1420 R=500
1430 CX=50
1440 CY=50
1450 CZ=50
1460 XC=500
1470 YC=500
1480 ZC=-2000
1500 DIM A(20),B(20)
1510 DIM X(20),Y(20),Z(20),S(20),F(20)
1520 RETURN
2000 *FX 4,1
2010 A$=INKEY$(0)
2020 IF A$="" THEN GOTO 2010
2040 IF A$="C" THEN R=R-10
2050 IF A$="F" THEN R=R+10
2060 IF ASC(A$)=136 THEN TH=TH-PI/180
2070 IF ASC(A$)=137 THEN TH=TH+PI/180
2080 IF ASC(A$)=139 THEN PH=PH+PI/180
2090 IF ASC(A$)=138 THEN PH=PH-PI/180
2100 PRINT TAB(0,28);"DISTANCE="
;R;TAB(15); "THETA=";INT(180*TH/PI)
;TAB(25)"PHI=";INT(180*PH/PI)
2110 IF A$="D" THEN RETURN
2120 GOTO 2010
3000 FOR I=0 TO P-1
3010 A(I)=-X(I)*ZC+Z(I)*XC
3020 B(I)=-Y(I)*ZC+Z(I)*YC
3025 W=Z(I)-ZC
3026 A(I)=A(I)/W
3027 B(I)=B(I)/W
3030 NEXT I
3040 RETURN
4000 CLS
4010 FOR I=0 TO L-1
4020 MOVE A(S(I)),B(S(I))
4030 DRAW A(F(I)),B(F(I))
4040 NEXT I
4050 RETURN
5000 CT=COS(TH)
5010 ST=SIN(TH)
5020 CP=COS(PH)
5030 SP=SIN(PH)
5040 TX=-(CX+R*CP*CT)
5050 TY=-(CY+R*SP)
5060 TZ=-(CZ+R*CP*ST)
5070 CX=-CP*ST/SQR(SP*SP+CP*CP*ST*ST)
5080 SX=-SP/SQR(SP*SP+CP*CP*ST*ST)
5090 CY=SQR(SP*SP+CP*CP*ST*ST)
5100 SY=-CP*CT
5110 RETURN
6000 FOR I=0 TO P-1
6005 IF I=P-1 THEN GOTO 6100
6010 X(I)=X(I)+TX
6020 Y(I)=Y(I)+TY
6030 Z(I)=Z(I)+TZ
6100 X=X(I)
6110 Y=Y(I)*CX-Z(I)*SX
6120 Z=Y(I)*SX+Z(I)*CX
6130 X(I)=X:Y(I)=Y:Z(I)=Z
6140 X=X(I)*CY-Z(I)*SY
6150 Y=Y(I)
6160 Z=X(I)*SY+Z(I)*CY
6200 X(I)=X:Y(I)=Y:Z(I)=Z
6300 NEXT I
6310 RETURN
7000 V=SQR(X(P-1)*X(P-1)+Y(P-1)*Y(P-1))
7010 C=Y(P-1)/V
7020 S=X(P-1)/V
7030 FOR I=0 TO P-2
7040 X=X(I)*C-Y(I)*S
7050 Y=X(I)*S+Y(I)*C
7060 X(I)=X+500:Y(I)=Y+500
7070 NEXT I
7080 RETURN
```

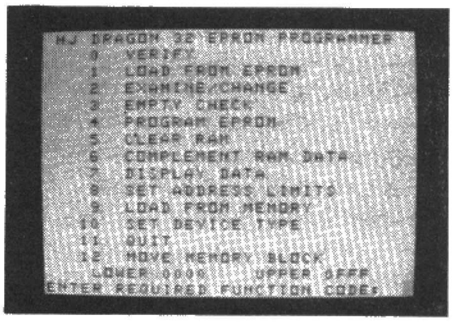
Dragon EPROM PROGRAMMER

Part 2 of Hew Jones Dragon EPROM programmer has full listings and description of the sophisticated operating software. Next month, a useful application example.

Because it is written in position independent code, the 4K machine code operating software can either be installed on the EPROM programmer card itself or loaded in from cassette when required. In the former (and much more desirable) instance there are two alternatives; either obtain a ready programmed 2732 or use the EPROM programmer to generate its own system firmware, which has a very definite appeal.

To accomplish this, enter the code given in the hex dump of **Listing 3** (incidentally, this was produced by the hard copy option of the list facility of the EPROM programmer itself) with a suitable Basic hex loader program, such as that of **Listing 2.** When finished, save to tape and re-load to &H4000, without the hardware installed, and try running at &H4002 to prove its correctness. All the screen formats, message texts and program flow integrity can be ascertained at this time.

After suitably amending any typing errors which may have occurred, the code can now be run with the EPROM programmer inserted and selected for 2732 operation. Use function 9 to load from $4000 onwards (check contents with function 7) and then blow a blank 2732 in function 4,



LISTING 3. hex dump.

```
      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0000 44 4B 4F 1F 8B 10 FF 0F A5 BE B4 4F 9F 72 B6 55  DKO........O.r.U
0010 97 71 30 8D 01 2F 17 0D 0D 30 8D 00 0D BF 01 E5  .q0../...0.....
0020 BE 04 20 9F 8B 0F 6F 17 0D 07 12 CC 10 00 FD 0F  .. ...o.........
0030 8D CC 2F FF FD 0F BF BE 0F 8D 4F 43 A7 80 BC 0F  ../.......OC....
0040 BF 26 F9 34 40 CE FF 40 4F A7 41 A7 43 A7 45 A7  .&.4@..@O.A.C.E.
0050 47 A7 44 43 A7 C4 A7 42 A7 46 B6 04 A7 41 A7 43  G.DC...B.F...A.C
0060 A7 45 A7 47 35 40 FF 0F AC 17 03 FB 17 03 09 17  .E.G5@..........
0070 02 E2 17 06 19 17 0A 98 12 17 03 EB BE 04 00 9F  ................
0080 8B 30 8D 00 C0 17 0C 9E 30 8D 00 D9 17 0C 97 30  .0......0......0
0090 8D 00 E1 17 0C 90 30 BD 00 F2 17 0C B9 30 8D 01  ......0......0..
00A0 02 17 0C B2 30 8D 01 0F 17 0C 7B 30 8D 01 1E 17  ....0.....{0....
00B0 0C 74 30 8D 01 29 17 0C 6D 30 8D 01 3E 17 0C 66  .t0..)..m0..>..f
00C0 30 8D 01 4C 17 0C 5F 30 BD 01 60 17 0C 58 30 8D  0..L.._0..`..X0.
00D0 01 72 17 0C 51 30 8D 06 20 17 0C 4A 30 BD 0B EA  .r..Q0.. ..J0...
00E0 17 0C 43 17 0A BB 17 05 DC 30 8D 01 6F 17 0C 36  ..C......0..o..6
00F0 17 0A D7 17 0B 2B 25 81 5D 10 27 FF 7C 1E 01 F7  .....+%.].'.|...

      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0100 0F A0 C1 0D 10 2A FF 71 34 04 17 03 5A 35 04 30  .....*.q4...Z5.0
0110 8D 00 12 34 10 30 8D 00 12 34 56 58 3A EC 84 E3  ...4.0...4VX:...
0120 62 ED 64 35 96 17 02 25 16 FF 4E 02 FB 01 BC 02  b.d5...%..N.....
0130 99 03 BB 04 33 05 DC 07 E0 06 F3 06 09 06 C7 08  ....3...........
0140 D0 08 8E 0B 5B 20 48 4A 20 44 52 41 47 4F 4E 20  ....[ HJ DRAGON
0150 33 32 20 45 50 52 4F 4D 20 50 52 4F 47 52 41 4D  32 EPROM PROGRAM
0160 4D 45 52 0D 00 20 20 20 20 30 20 20 56 45 52 49  MER..    0  VERI
0170 46 59 0D 00 20 20 20 20 31 20 20 4C 4F 41 44 20  FY..    1  LOAD
0180 46 52 4F 4D 20 45 50 52 4F 4D 0D 00 20 20 20 20  FROM EPROM..
0190 32 20 20 45 58 41 4D 49 4E 45 2F 43 48 41 4E 47  2  EXAMINE/CHANG
01A0 45 0D 00 20 20 20 20 33 20 20 45 4D 50 54 59 20  E..    3  EMPTY
01B0 43 48 45 43 4B 0D 00 20 20 20 20 34 20 20 50 52  CHECK..    4  PR
01C0 4F 47 52 41 4D 20 45 50 52 4F 4D 0D 00 20 20 20  OGRAM EPROM..
01D0 20 35 20 20 43 4C 45 41 52 20 52 41 4D 0D 00 20   5  CLEAR RAM..
01E0 20 20 20 36 20 20 43 4F 4D 50 4C 45 4D 45 4E 54     6  COMPLEMENT
01F0 20 52 41 4D 20 44 41 54 41 0D 00 20 20 20 20 37   RAM DATA..    7

      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0200 20 20 44 49 53 50 4C 41 59 20 44 41 54 41 0D 00    DISPLAY DATA..
0210 20 20 20 20 20 38 20 20 53 45 54 20 41 44 44 52       8  SET ADDRE
0220 53 53 20 4C 49 4D 49 54 53 0D 00 20 20 20 20 39  SS LIMITS..    9
0230 20 20 4C 4F 41 44 20 46 52 4F 4D 20 4D 45 4D 4F    LOAD FROM MEMO
0240 52 59 0D 00 20 20 20 31 30 20 20 53 45 54 20 44  RY..   10  SET D
0250 45 56 49 43 45 20 54 59 50 45 0D 00 45 4E 54 45  EVICE TYPE..ENTE
0260 52 20 52 45 51 55 49 52 45 44 20 46 55 4E 43 54  R REQUIRED FUNCT
0270 49 4F 4E 20 43 4F 44 45 3A 20 00 0D 20 4C 4F 41  ION CODE: .. LOA
0280 44 49 4E 47 20 44 41 54 41 20 46 52 4F 4D 20 4D  DING DATA FROM M
0290 41 53 54 45 52 20 45 50 52 4F 4D 20 0D 00 20 20  ASTER EPROM ..
02A0 20 20 20 4F 50 45 52 41 54 49 4F 4E 20 43 4F 4D     OPERATION COM
02B0 50 4C 45 54 45 44 20 0D 00 20 20 20 20 45 58 41  PLETED ..    EXA
02C0 4D 49 4E 45 20 4D 45 4D 4F 52 59 0D 20 45 4E 54  MINE MEMORY. ENT
02D0 45 52 20 53 54 41 52 54 49 4E 47 20 41 44 44 52  ER STARTING ADDR
02E0 45 53 53 20 20 3A 00 17 01 7D 30 8C 8E 17 0A 36  ESS  :...}0....6
02F0 17 0B AB 17 00 0C 17 00 1D 17 00 21 17 00 2C 26  ...........!..,&

      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0300 F2 39 17 00 07 17 07 E9 17 08 05 39 FC 0F 91 F7  .9.........9....
0310 FF 40 B7 FF 42 39 B6 0F 9E B7 0F 9F 39 FC 0F BD  .@..B9......9...
0320 F3 0F 91 1E 01 B6 0F 9F A7 84 39 BE 0F 91 B6 FF  ..........9.....
0330 30 01 BF 0F 91 BC 0F 93 23 06 30 1F BF 0F 91 43  0.......£.0....C
0340 4D 39 BE 0F 91 27 05 30 1F BF 0F 91 39 30 BD FF  M9...'.0....90..
0350 4D 17 09 D2 17 07 B9 B6 08 B1 0F A0 27 11 8E 00  M...........'...
0360 00 BF 0F 91 FC 0F BF 1C FE B3 10 00 FD 0F 93 C6  ................
0370 0A 17 00 3E 5A 26 FA 39 9E 8A BF 0F 91 7F 0F 9D  ...>Z&.9........
0380 17 00 07 17 06 7C 17 00 01 39 B6 0F 9D 30 BD 00  .....|...9...0..
0390 15 1F B9 58 3A EC 84 FD 0F 93 FD 0F 9B 1C FE C3  ...X:...........
03A0 10 00 FD 0F 8F 39 07 FF 0F FF 07 FF 0F FF 0F FF  .....9..........
03B0 1F FF 34 07 86 32 20 02 34 07 C6 B0 5A 26 FD 4A  ..4..2 .4...Z&.J
03C0 26 F8 35 87 17 00 A0 30 8D FE EE 17 09 5B 17 07  &.5....0.....X..
03D0 F9 17 07 FF 25 EE 5D 27 41 BC 0F 93 22 E6 BF 0F  ....%.]'A..."...
03E0 91 17 00 D3 BE 05 04 9F 88 17 00 9E B6 0F 9F 17  ................
03F0 00 D3 17 09 5B 81 0D 27 19 81 48 27 1D 81 5E 27  ....[..'..H'..^'

      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0400 1D B1 0A 27 10 17 00 45 25 D7 17 00 CB B7 0F 9F  ...'...E%.......
0410 20 D2 17 FF 0B 17 FF 13 26 C7 17 07 81 39 17 FF   .......&....9..
0420 21 20 BE 17 00 41 30 8D 00 4D 17 FE D5 17 00 87  ! ...A0..M......
0430 B6 0F 9E B1 0F 9F 26 09 17 FE F0 26 ED 17 00 65  ......&....&...e
0440 39 17 FE D2 12 17 00 FC 17 00 E6 20 EB 1C FE 80  9.......... ....
0450 30 25 10 81 09 23 0D 80 07 81 0A 25 06 81 0F 23  0%...£.....%...£
0460 03 1A 01 39 1C FE 39 34 13 BE 04 00 B6 60 A7 80  ...9..94.....`..
0470 8C 06 00 26 F9 35 93 0D 56 45 52 49 46 59 20 4F  ...&.5..VERIFY O
0480 50 45 52 41 54 49 4F 4E 0D 00 12 17 05 FE 17 05  PERATION........
0490 FB BE 0F 91 17 07 2A 17 05 F2 39 41 44 44 52 45  ......*...9ADDRE
04A0 53 53 20 0D 00 30 8D 00 04 17 08 7A 39 20 50 41  SS ..0.....z9 PA
04B0 53 53 45 44 20 0D 00 FC 0F 91 C3 10 00 1F 01 A6  SSED ...........
04C0 84 B7 0F 9F 39 17 05 C4 17 05 C1 17 05 BE BE 0F  ....9...........
04D0 9F 17 06 D1 17 05 B5 39 84 0F 34 02 B6 0F 9F 48  .......9..4....H
04E0 48 48 48 AA E0 39 17 FF 7E 30 8D 00 2C 17 08 36  HHH..9..~0..,..6
04F0 17 FE 0F B6 FF B1 0F 9E 26 0D 17 FE 2E 26 F1 30  ........&....&.0

      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0500 8D 00 26 17 08 20 39 17 FE 0C 17 00 37 17 00 21  ..&.. 9.....7..!
0510 20 EB 20 44 41 54 41 20 00 0D 20 45 4D 50 54 59   . DATA .. EMPTY
0520 20 43 48 45 43 4B 20 0D 00 20 50 41 53 53 20 0D   CHECK .. PASS .
0530 00 17 08 1C 81 48 27 05 81 0D 26 F5 39 10 FE 0F  .....H'...&.9...
0540 A5 16 FB 2B 30 8D 00 B8 17 07 DB 17 05 3E BE 0F  ...+0........>..
0550 91 17 06 6D 17 05 35 17 FF 6B 17 06 41 39 30 8D  ...m..5..k..A90.
0560 00 AA 17 07 C1 BE 0F 91 BF 0F A1 17 03 8E 17 05  ................
0570 45 B6 10 B7 0F A4 17 01 0C 7A 0F A4 26 25 BE 05  E........z..&%..
0580 40 9F B8 17 05 06 BE 0F 91 17 06 35 17 04 FD FC  @..........5....
0590 0F 91 B6 20 C5 10 27 02 B6 2A AD 9F A0 02 B6 10  ... ..'..*......
05A0 B7 0F A4 17 FF 11 12 B6 0F 9F 4C 27 0B 17 03 54  ..........L'...T
05B0 17 00 FB 17 05 0E 26 28 17 FD 70 26 BC 17 03 40  ......&(..p&...@
05C0 17 04 F3 17 00 CB BE 0F A1 BF 0F 91 30 BD 00 5B  ............0..X
05D0 17 07 53 17 00 AB 17 05 37 17 00 A5 17 FE 4B 39  ..S.....7.....K9
05E0 17 03 1D 17 04 D0 17 00 A5 30 BD 00 5E 17 07 36  .........0..^..6
05F0 17 00 8E 17 05 1A 17 FD 1D 17 FF 48 17 FF 32 39  ...........H..29

      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0600 20 46 41 49 4C 45 44 20 41 54 20 00 0D 20 45 50   FAILED AT .. EP
0610 52 4F 4D 20 50 52 4F 47 52 41 4D 20 4F 50 45 52  ROM PROGRAM OPER
0620 41 54 49 4F 4E 20 0D 00 0D 20 43 4F 4D 50 4C 45  ATION ... COMPLE
0630 54 45 44 20 43 4F 50 59 49 4E 47 2C 43 48 45 43  TED COPYING,CHEC
0640 4B 49 4E 47 20 4E 4F 57 20 0D 00 20 46 41 49 4C  KING NOW .. FAIL
0650 45 44 20 50 52 4F 47 52 41 4D 20 56 45 52 49 46  ED PROGRAM VERIF
0660 49 43 41 54 49 4F 4E 20 21 0D 20 50 52 4F 47 52  ICATION !. PROGR
0670 41 4D 4D 49 4E 47 20 41 42 4F 52 54 45 44 20 0D  AMMING ABORTED .
0680 00 C6 0A 20 02 C6 28 17 FD 28 5A 26 FA 39 B6 FF  ... ..(..(Z&.9..
0690 46 84 3F 8A 00 B7 FF 46 8D 01 39 34 02 B6 FF 45  F.?....F..94...E
06A0 84 C7 8A 30 B7 FF 45 8A 08 B7 FF 45 35 82 17 FC  ...0..E....E5...
06B0 5B 17 04 02 B6 0F 9E B7 FF 44 17 03 D9 17 FC F2  [.........D.....
06C0 17 03 F3 12 39 30 BD 00 1A 17 06 5A BE 0F 91 17  ....90.....Z....
06D0 04 EF 30 8D 00 18 17 06 4D BE 0F 93 17 04 E2 17  ..0.....M.......
06E0 04 BC 39 20 20 20 20 4C 4F 57 45 52 20 00 20 20  ..9    LOWER .
06F0 20 20 55 50 50 45 52 20 00 20 20 20 31 31 20 20    UPPER .   11
```

LISTING 1. BASIC test program.

```
10 CLS
20 PRINT "TEST 1-1.6821 OUTPUT TEST"
30 PRINT"SQUARE WAVE ON PORTS A/B"
40 POKE &HFF41,00:POKE &HFF43,00
50 P1=&HFF40:P2=&HFF44:R2=&HC000
60 POKE P1+0,255:POKE P1+2,255
70 POKE P1+1,4:POKE P1+3,4
80 PRINT "HIT ANY KEY TO STOP TEST"
90 I=0
100 A$=INKEY$
110 IF A$<>"" THEN 195
120 POKE P1+0,I:POKE P1+2,255-I
130 I=I+1
140 IF I>255 THEN I=0
150 GO TO 100
195 CLS
200 PRINT "TEST 2-2.6821 PORT A O/P "
210 PRINT "SQUARE WAVE ON PORT A"
220 POKE P2+1,00:POKE P2+0,255
230 POKE P2+1,4
240 I=0
250 PRINT"HIT ANY KEY TO STOP TEST"
260 A$=INKEY$
270 IF A$<>"" THEN 400
280 POKE P2+0,I
290 I=I+1
300 IF I>255 THEN I=0
310 GO TO 260
400 CLS:PRINT"TEST 3-2.6821 PORT A INPUT"
410 POKE P2+1,0:POKE P2+0,0:POKE P2+1,4
420 PRINT "HIT ANY KEY TO STOP TEST"
430 PRINT@162,"DATA ";HEX$(PEEK(P2+0));
440 A$=INKEY$
450 IF A$<>"" THEN 500
460 GO TO 430
500 CLS:PRINT" TEST 4-RELAY TEST"
505 POKE P2+3,&H0:POKE P2+2,255:POKE P2+3,&H4
506 POKE P2+1,&H34:REM CA2 LOW
510 PRINT"CODE 0-7 ENABLES RELAYS"
515 PRINT"RELAY CODE GREATER THAN 7 STOPS TEST"
520 INPUT" RELAY CODE :- ";RC
525 IF RC>7THEN 600
530 D=RC AND &H07
540 POKE P2+2,D
550 GO TO 520
600 CLS:PRINT"TEST 5-VOLTAGE TEST VPP"
610 PRINT"VOLTAGE CODE ABOVE 3 STOPS TEST"
620 POKE P2+2,0:REM RELAYS OFF
630 PRINT@160,"VOLTAGE CODES ARE :-"
640 PRINT@192," 0=5V  1=25V  2=6V  3=21V"
650 INPUT"DESIRED VOLTAGE";RC
660 IF RC<0 OR RC>3 THEN 800
670 RC=RC*64:REM SHIFT LEFT
680 POKE P2+2,RC
690 POKE P2+1,&H3C:POKE P2+1,&H34
710 GO TO 630
800 PRINT "TEST 6-MEMORY TEST"
810 FOR I=0 TO &HFFF
812 PRINT HEX$(R2+I);" ";
814 FOR J=0 TO 3
820 PRINT HEX$(PEEK(R2+I));" ";
822 I=I+1
824 NEXT J
826 PRINT " "
828 I=I-1
830 NEXT I
840 END
```

which can then be installed on the card. From now on, when power is first applied the unit auto-boots although BASIC may be entered at any time. The reset button reverts to BASIC mode.

The software reserves RAM from $1000-$2FFF for the 'map', and uses $F80-FFF for variables and scratchpad. Calls to the Basic interpreter ROMs have been kept to an obsolute minimum and therefore the EPROM programmer should run with a minor software modification on the Dragon 64 and a Colour Computer with extended Basic. The Basic 'hot' start vector for your machine must be substituted at $C00A,$C00B. This can be established by PEEKing the contents of $72,$73 respectively.

Summary

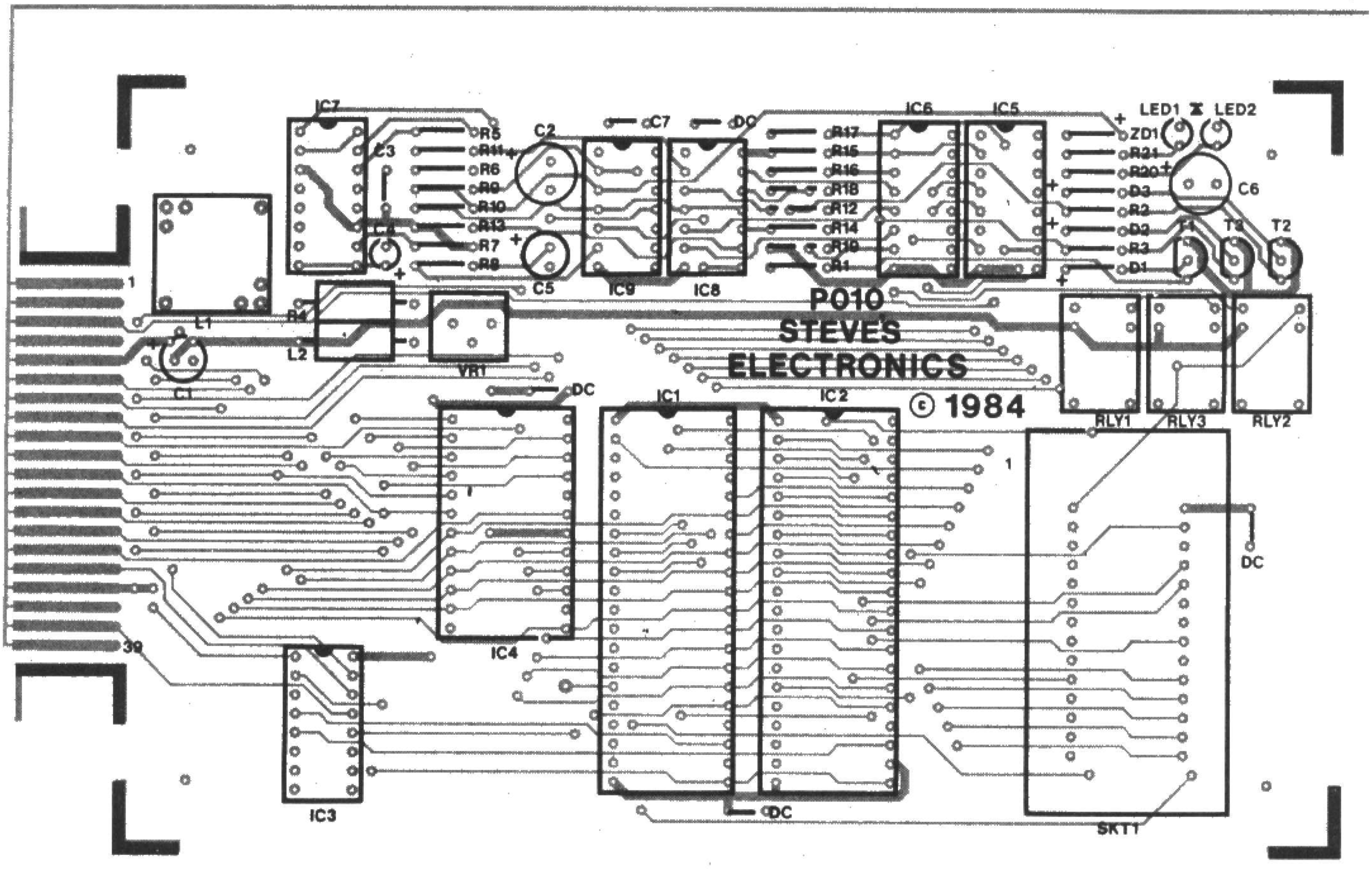
In the course of testing the design, the card successfully programmed all the supported devices, most of which were obtained by shameless begging and borrowing. These included 2764s manufactured by Hitachi, Fujitsu, Mitsubishi, Intel and AMD. Unfortunately two factors prevented the design from accommodating 27128s and intelligent programming; lack of board space and the limited extra capacity available from the power supply.

Steve's Electronics, Castle Arcade, Cardiff (Tel: 0222 41905) have kindly agreed to stock and supply any or all of the components in this article including the prototyping board, coil winding kit and ready programmed EPROM. ▶

LISTING 2. BASIC hex loader.

```
5 CLS
10 INPUT "STARTING ADDRESS IN HEX";S$
15 INPUT "END ADDRESS IN HEX";E$
20 S1$="&H"+S$
25 E1$="&H"+E$
30 S=VAL(S1$)
35 E=VAL(E1$)
40 PRINT@480,HEX$(S);" ";HEX$(PEEK(S));"
50 INPUT "NEW DATA";D$
55 D1$="&H"+D$
60 D=VAL(D1$)
70 POKE S,D
80 S=S+1
90 IF S<=E THEN 40
100 END
```

```
     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0700 51 55 49 54 20 0D 00 30 BD 00 11 17 06 18 86 FF  QUIT ..0........
0710 B7 0F 9F 17 FC 07 17 FC 12 26 F3 39 0D 20 43 4C  .........&.9. CL
0720 45 41 52 20 52 41 4D 20 4F 50 45 52 41 54 49 4F  EAR RAM OPERATIO
0730 4E 20 0D 00 30 8D 00 5B 17 05 EB 17 FF 87 30 8D  N ..0..[......0.
0740 00 6C 17 05 E1 BE 0F 91 BF 0F A1 30 8C 95 17 05  .l.........0....
0750 D5 17 04 76 17 04 7C 25 EC 5D 27 26 BF 0F 91 BC  ...v..|%.]'&....
0760 0F 9B 22 1F 30 BC B7 17 05 BC 17 04 5D 17 04 63  ..".0.......]..c
0770 25 D3 5D 27 0D BC 0F 91 25 09 BC 0F 9B 22 04 BF  %.]'....%...."..
0780 0F 93 39 30 8D 00 42 17 05 9C BE 0F A1 BF 0F 91  ..90..B.........
0790 16 FF B2 0D 20 53 45 54 20 4E 45 57 20 41 44 44  .... SET NEW ADD
07A0 52 45 53 53 20 4C 49 4D 49 54 53 20 0D 00 20 45  RESS LIMITS .. E
07B0 4E 54 45 52 20 4E 45 57 20 41 44 44 52 45 53 53  NTER NEW ADDRESS
07C0 20 52 41 4E 47 45 20 0D 00 0D 20 49 4E 56 41 4C   RANGE ... INVAL
07D0 49 44 20 41 44 44 52 45 53 53 20 47 49 56 45 4E  ID ADDRESS GIVEN
07E0 20 2D 20 52 45 50 45 41 54 20 45 4E 54 52 59 20   - REPEAT ENTRY
07F0 0D 00 34 40 30 BD 01 4F 17 05 2B 30 BD 01 6A 17  ..4@0..O..+0..j.

     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0800 05 24 17 03 C5 17 03 CB 25 EB 5D 27 0F 1E 13 A6  .$......%.]'....
0810 C0 B7 0F 9F 17 FB 06 17 FB 11 26 F3 35 C0 34 40  ...........&.5.4@
0820 30 BD 01 63 17 04 FF 7F 0F A7 30 BD 01 70 17 04  0..c......0..p..
0830 F5 17 03 96 17 03 9C 25 E5 5D 27 7C 1F 10 C4 F0  .......%.]'|....
0840 1F 01 BF 0F 91 30 8D 04 C3 17 04 DA 17 05 01 81  .....0..........
0850 59 26 17 C6 10 F7 0F A9 F7 0F A8 F7 0F A7 C6 FE  Y&..............
0860 D7 6F CC 01 00 FD 0F AA 20 12 C6 10 F7 0F A9 C6  .o...... .......
0870 04 F7 0F A8 CC 00 40 FD 0F AA 0F 6F 17 03 1F BE  ......@....o....
0880 0F 91 34 10 17 02 CE F6 0F A9 17 03 11 FE 0F 91  ..4.............
0890 17 01 32 FF 0F 91 17 02 B6 5A 26 EE 35 10 BF 0F  ..2......Z&.5...
08A0 91 7D 0F A7 27 03 17 02 F5 17 04 A4 81 0A 27 15  .}..'.........'.
08B0 B1 5E 27 22 81 48 26 F1 0F 6F 7F 0F A7 17 04 71  .^'".H&..o.....q
08C0 17 02 DB 35 C0 FC 0F AA F3 0F 91 10 B3 0F 9B 24  ...5...........$
08D0 DB FD 0F 91 20 A6 FC 0F 91 B3 0F AA FD 0F 91 24  .... ..........$
08E0 9B CC 00 00 FD 0F 91 16 FF 92 F6 FF 45 C4 FB F7  ............E...
08F0 FF 45 12 B7 FF 44 CA 04 F7 FF 45 39 86 FF 20 EA  .E...D....E9.. .

     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0900 86 00 20 E6 B6 0F 9F B7 0F 9E 39 30 8D 00 16 17  .. .......90....
0910 04 14 17 FD A2 B6 0F 9F 43 B7 0F 9F 17 F9 FE 17  ........C.......
0920 FA 09 26 EE 39 0D 20 43 4F 4D 50 4C 45 4D 45 4E  ..&.9. COMPLEMEN
0930 54 20 52 41 4D 20 44 41 54 41 20 4F 50 45 52 41  T RAM DATA OPERA
0940 54 49 4F 4E 20 0D 00 0D 20 4C 4F 41 44 20 4D 41  TION ... LOAD MA
0950 50 20 46 52 4F 4D 20 49 4E 54 45 52 4E 41 4C 20  P FROM INTERNAL
0960 4D 45 4D 4F 52 59 20 0D 00 20 53 50 45 43 49 46  MEMORY .. SPECIF
0970 59 20 4C 4F 41 44 20 53 4F 55 52 43 45 20 41 44  Y LOAD SOURCE AD
0980 44 52 45 53 53 20 00 20 4D 45 4D 4F 52 59 20 44  DRESS . MEMORY D
0990 55 4D 50 20 43 4F 4D 4D 41 4E 44 20 0D 00 20 45  UMP COMMAND .. E
09A0 4E 54 45 52 20 4C 49 53 54 20 53 54 41 52 54 20  NTER LIST START
09B0 41 44 44 52 45 53 53 20 00 4F 10 FE 0F A5 FE 0F  ADDRESS .O......
09C0 AE 6E 9F FF FE 34 56 FC 0F 91 10 B3 0F 93 24 29  .n...4V.......$)
09D0 17 00 B9 12 12 BE 0F 91 17 01 E6 17 00 AE F6 0F  ................
09E0 A8 34 04 17 FA D1 BE 0F 9F 17 01 B9 17 00 9D 17  .4..............
09F0 F9 39 35 04 27 03 5A 26 E8 35 D6 30 8D 00 29 17  .95.'.Z&.5.0..).

     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0A00 03 24 17 01 99 30 8D 00 3A 17 03 1A 17 01 BB 17  .$...0..:.......
0A10 02 0F 25 EE 8C 00 06 24 E9 1F 10 F7 0F 9D 17 00  ..%....$........
0A20 EF 17 F9 66 17 FC 67 39 0D 20 53 50 45 43 49 46  ...f..g9. SPECIF
0A30 59 20 54 48 45 20 45 50 52 4F 4D 20 54 59 50 45  Y THE EPROM TYPE
0A40 20 0D 00 20 45 4E 54 45 52 20 44 45 56 49 43 45   .. ENTER DEVICE
0A50 20 43 4F 44 45 20 3A 20 0D 20 30 2D 32 37 31 36   CODE : . 0-2716
0A60 20 31 2D 32 37 33 32 20 32 2D 32 35 31 36 20 20   1-2732 2-2516
0A70 20 0D 20 33 2D 32 35 33 32 20 34 2D 32 37 33 32   . 3-2532 4-2732
0A80 41 20 35 2D 32 37 36 34 20 20 0D 00 34 02 86 20  A 5-2764  ..4..
0A90 AD 9F A0 02 35 82 34 14 30 8D 00 14 F6 0F 9D 3A  ....5.4.0......:
0AA0 E6 84 C1 FF 27 0B F7 FF 46 17 FB EF 1C FB 35 94  ....'...F.....5.
0AB0 58 46 58 41 C6 D4 34 14 30 BD 00 02 20 DE 50 4E  XFXA..4.0... .PN
0AC0 50 51 CE F4 17 FE 39 34 14 30 8D 00 1E BD 3C 35  PQ....94.0....<5
0AD0 14 27 13 B6 FF 44 34 02 8D DC 17 FE 1F 35 02 B7  .'...D4......5..
0AE0 0F 9E B1 0F 9F 39 17 FE 13 4F 39 40 FF 40 FF FF  .....9...O9@.@..
0AF0 E4 17 FE 0C 34 14 30 BD 00 0B BD 0F 35 14 B6 FF  ....4.0.....5...

     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0B00 44 B7 0F 9E 39 00 04 00 01 04 24 32 7D 16 FF BC  D...9.....$2}...
0B10 34 14 30 8D 00 03 16 FF B3 18 0C 18 11 0C 2C 34  4.0...........,4
0B20 16 FC 0F 91 10 B3 0F 93 24 29 17 FF 5F F6 0F A8  ........$)._....
0B30 34 04 17 F9 82 B6 2E F6 0F 9F C1 20 25 07 C1 7F  4.......... %...
0B40 24 03 B6 0F 9F AD 9F A0 02 17 F7 DF 35 04 27 03  $...........5.'.
0B50 5A 26 DD 35 96 34 16 7D 0F A7 27 35 17 00 3F C6  Z&.5.4.}..'5..?.
0B60 06 34 04 17 FF 26 35 04 5A 26 F6 F6 0F A8 7F 0F  .4...&5.Z&......
0B70 A3 39 04 86 2D AD 9F A0 02 B6 0F A3 17 00 14 AD  .4..-...........
0B80 9F A0 02 17 FF 06 35 04 7C 0F A3 5A 26 E3 17 00  ......5.|..Z&...
0B90 0D 35 16 84 0F 81 09 23 02 8B 07 8B 30 39 86 0D  .5.....£....09..
0BA0 AD 9F A0 02 39 34 14 A6 B4 34 02 44 44 44 44 17  ....94...4.DDDD.
0BB0 FF E1 AD 9F A0 02 35 02 17 FF D8 AD 9F A0 02 35  ......5........5
0BC0 94 17 FF E1 30 01 17 FF DC 39 17 01 95 30 01 BF  ....0....9...0..
0BD0 0F 97 39 BE 0F 97 10 BE 0F 80 A6 B0 27 0D 81 0D  ..9.........'...
0BE0 27 09 17 F8 68 25 32 A7 A0 20 EF 6F A4 6A A4 9E  '...h%2.. .o.j..
0BF0 BA 10 BE 0F 80 1F 10 A6 A4 4C 27 18 1F 10 1C FE  .........L'.....

     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0C00 59 49 1C FE 59 49 1C FE 59 49 1C FE 59 49 EA A0  YI..YI..YI..YI..
0C10 1F 01 20 E3 C6 FF 1C FE 39 5F A6 80 26 FB 1A 01  .. .....9_..&...
0C20 39 BE 0F 97 4F 5F FD 0F BB FD 0F 8B FD 0F B9 A6  9...O_..........
0C30 B0 27 1B 81 0D 27 17 17 00 3E 10 26 FF DB 34 02  .'...'...>.&..4.
0C40 17 00 13 35 04 4F F3 0F BB FD 0F 8B 20 E1 BE 0F  ...5.O...... ...
0C50 8B C6 FF 1C FE 39 34 06 86 0A F6 0F BC 3D FD 0F  .....94......=..
0C60 B9 86 0A F6 0F BB 3D FB 0F B9 89 00 12 FD 0F BB  ......=.........
0C70 FC 0F 89 FD 0F BB 35 86 B0 30 25 07 B1 39 22 03  ......5..0%..9".
0C80 1A 04 39 1C FB 39 30 BD 00 6B 17 00 99 17 FA B5  ..9..90..k......
0C90 30 8D 00 50 17 00 8F 17 FF 30 17 FF 36 25 2A BC  0..P.....0..6%*.
0CA0 0F 9B 24 E2 BF 0F 95 FC 0F 91 C3 10 00 1F 01 FC  ..$.............
0CB0 0F 95 C3 10 00 1F 02 FC 0F 93 C3 10 00 FD 0F 8B  ................
0CC0 A6 80 A7 A0 BC 0F 88 26 F7 39 20 20 20 31 32 20  .......&.9   12
0CD0 20 4D 4F 56 45 20 4D 45 4D 4F 52 59 20 42 4C 4F   MOVE MEMORY BLO
0CE0 43 4B 20 00 20 44 45 53 54 49 4E 41 54 49 4F 4E  CK . DESTINATION
0CF0 20 00 20 20 20 20 4D 45 4D 4F 52 59 20 4D 4F 56   .    MEMORY MOV

     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F

0D00 45 20 43 4F 4D 4D 41 4E 44 20 0D 00 20 20 20 20  E COMMAND ..
0D10 48 41 52 44 20 43 4F 50 59 20 52 45 51 55 49 52  HARD COPY REQUIR
0D20 45 44 20 3F 20 00 A6 80 27 06 AD 9F A0 02 20 F6  ED ? ...'..... .
0D30 39 34 16 0F 6E 96 6F 27 06 9E 99 DC 9B 20 09 D6  94..n.o'..... ..
0D40 89 94 1F 8E 10 10 86 20 9F 6A D7 6C 97 6D 35 96  ....... .j.l.m5.
0D50 34 14 BD B0 09 BD B0 06 27 FB C6 60 E7 9F 00 88  4.......'..`....
0D60 35 94 0F 87 8E 02 DD C6 01 17 FF E4 81 08 26 07  5.............&.
0D70 5A 27 EF 30 1F 20 20 81 0D 26 0D 4F 34 01 17 FE  Z'.0.  ..&.O4...
0D80 1D 6F 84 8E 02 DC 35 81 81 20 25 DD 81 7B 24 D9  .o....5.. %..{$.
0D90 C1 FA 24 D5 A7 B0 5C AD 9F A0 02 20 CC 35 32 2C  ..$...\.... .52,
0DA0 50 43 52 20 3B 54 49 54 4C 45 0D 20 4C 42 53 52  PCR ;TITLE. LBSR
0DB0 20 50 52 49 4E 54 4D 0D 20 4C 42 53 52 20 46 38   PRINTM. LBSR F8
0DC0 4C 50 31 20 3B 4C 4F 57 45 52 2F 55 50 50 45 52  LP1 ;LOWER/UPPER
0DD0 20 4C 49 4D 49 54 53 0D 20 4C 45 41 58 20 4D 4E   LIMITS. LEAX MN
0DE0 4F 35 31 2C 50 43 52 20 3B 47 45 54 20 44 45 53  O51,PCR ;GET DES
0DF0 54 49 4E 41 54 49 4F 49 4E 20 41 44 44 52 45 53  TINATIOIN ADDRES
```



Overlay of the Dragon EPROM blower's PCB.

BYTE BACK

Send your letters to The Editor, E&CM, 155 Scriptor Court, Farringdon Road, London EC1R 3AD.

Sir,
Having built up the Rom boards, I find that they will not run **all** manufacturers' Eproms, contrary to page 19 of the article.

I have found that if the main board is an issue 4 or above, all the Eproms function correctly. However, on issue 3 boards there are problems. I cannot read Intel chips, although they work perfectly in the main board. The problems experienced range from refusal to recognise a chips' presence; or hangs the machine either on entry or during execution.

An odd thing is that one appears to get further into the program when 10nF capacitors are fitted than one does with 100nF.

Lead lengths (ribbon cable) are kept at approximately 10". The Intel chips are standard speed, 200/250ns. To check that chip timings weren't causing the problems, Hitachi 250ns and 450ns along with Fuji at 300ns were tried with no problems.

I have also tried fitting a 10μF capacitor onto the board, but to no avail.

Our experience is that machines fitted with issue 3 boards all experience the problems as highlighted above, but those machines with issue 4 and above have experienced no problems, performing satisfactorily, with all 3 makes of Eproms.

Trusting you can highlight the areas of the problems.

Mr. A. Wiseman
Tamworth

There are two possible sources of the problem. Firstly, the ROM boards have been proved to operate successfully on issue 3 boards. It may be that the Intel chips you are using are too slow – Acorn recommend a minimum of 300ns, but 450ns is the optimum speed. The reason why the chips work on the main board but not on the ROM board is the existence of a buffer chip between the BBC and the ROM board. I would therefore recommend the use of faster chips. However, you could also be suffering from the 'dirty power supply problem', ie oxidised connectors which reduce the supply to less than the required 5V. Cleaning the connectors will rectify this.

Sir,
As a regular reader for some months now I would like to congratulate you on the standard you have achieved. I find the articles generally interesting, neither so technical as to be bewildering nor patronising.

What I find particularly appealing however is your inclusion of software to actually use the projects once built, without much research, as hardware and software are mutually supportive.

A major hurdle in building any electronics project is actually obtaining the components specified especially when purchasing by post. Some of the parts required never seem to appear in adverts, and others only in variant forms. For example, in the ZX Spectrum real time clock project (April/May 1984) one of the components required is a 4040 BE. Is this the same as a 4040 or is there an equivalent component? In the same project a 6116 CMOS RAM is specified – but what speed?

The lack of information such as this is sufficient to prevent many people from attempting such projects.

This could be remedied to a great extent by giving more information on components and suggesting sources – please bear in mind not all of us live near Watford.

D. Bradbeer
Edinburgh

Point taken. Component supply is becoming a problem and we will endeavour to publish sources of the scarcer components in future issues (and yes, a 4040 is the same as a 4040 BE).

Sir,
It has come to my attention that the Speech Synthesiser board I designed for Jan/Feb '84 of your magazine does not work correctly with issue 2&3 Spectrums. The fault manifests itself as the synthesiser running through the allophones too quickly, thus garbling the speech.

This fault can be corrected with a small software patch:-

The line of Basic, in each of the example programs which waits for the synthesiser to become 'not busy' must be changed from:

IF IN 159 > 127 THEN GOTO . . .

To:

IF IN 159 > 127 **THEN PAUSE 1:** GOTO . . .

The added Pause cures the problem, which seems due to a timing difference between issue 1 Spectrums (the project was developed on an Issue 1) and the newer Issue Spectrums.

Robert Harvey
Oxford

Sir,
Having read the advance publicity for Adam Denning's article on inhibiting the action of the BREAK key on a BBC micro, I was disappointed in the event to find that is was no more than an interception of the break code using *FX247,8 and 9 (clever though that is!). The problem is that this requires individual tailoring of each separate program that you wish to protect, as described in the article, so that control can be recovered at a suitable place in the program. What would be really desirable would be complete disabling of the BREAK key so that its action would be ignored.

At the school where I teach we have a large Econet installation. We have adopted an idea originally suggested by Felsted School, which enables software control of the BREAK key on all the machines connected to the network, and we have developed a control program which enables combinations of the Screen, Keyboard and BREAK key to be selectively disabled. The status quo can later be restored, and any program running continues to do so while the disabling is in force.

The technique involves a small hardware modification to the BBC Micro, and so it should only be attempted by experienced users. It will also invalidate any remaining guarantee. Pin 4 of IC16 which is in the NW corner of the PCB is carefully cut and bent horizontal. This pin is the reset pin of the 555 timer which produces a pulse to reset the system when the BREAK key is pushed. It is fed via a single stage amplifier from pin 11 of IC7, (the serial ULA to the centre rear of the PCB) which controls the cassette relay. Thus whenever the cassette relay is energised the BREAK key is disabled. This is not usually a problem if using cassette, as the ESCAPE key can be used to stop a program loading or saving and turn the cassette motor off again. Thus with the mod installed *M.1 (or *FX137,1) will disable the BREAK key and *M.0 (or *FX137,0) will re-enable it.

IC16 pin 4 is cut through and bent up horizontally. It is attached to the collector of the BC108.

The connection to IC7 pin 11 can conveniently be made to the positive hole marked / or D9 to the left of IC7 as D9 is not fitted.

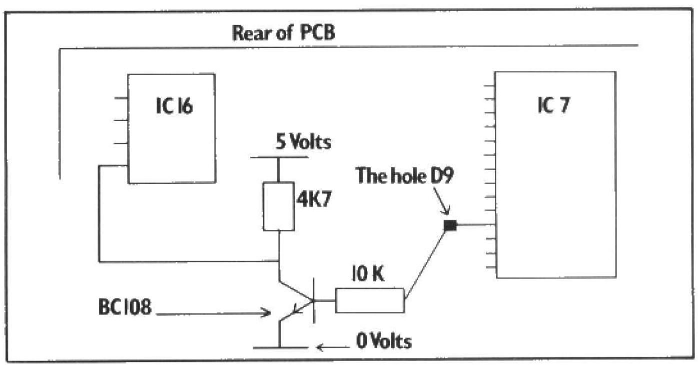
±5v is obtainable at many places on the board.

On an issue 3 board the extra components can be fitted in the holes allocated to S3 and IC90 which are not used in the ECONET upgrade.

For more recent machines the components can be fitted on a small scrap of veroboard attached to the side of the case near the power supply with double sided adhesive pads.

I trust that this alternative solution will be of interest to your readers!

Mr. R. Newman
The Microelectronics Centre
Peterborough



Sir,
I was impressed with your computer Burglar Alarm circuit which was included with April's *E&CM*.

While the project is marvellously simply and provides an ideal introduction to peripherals, I feel that the computers shown were a little high powered.

It seems a bit like overkill to use a machine as powerful as the BBC to monitor a few sensors.

I would have thought a more obvious choice would be the ZX81 or more importantly the now very cheap multitasking Jupiter Ace. Both these machines are cheap enough to be bought specifically for the task and could be dedicated to monitoring complex alarm systems.

With a little imagination from the constructor/programmer several different actions could be initiated from different alarm sources.

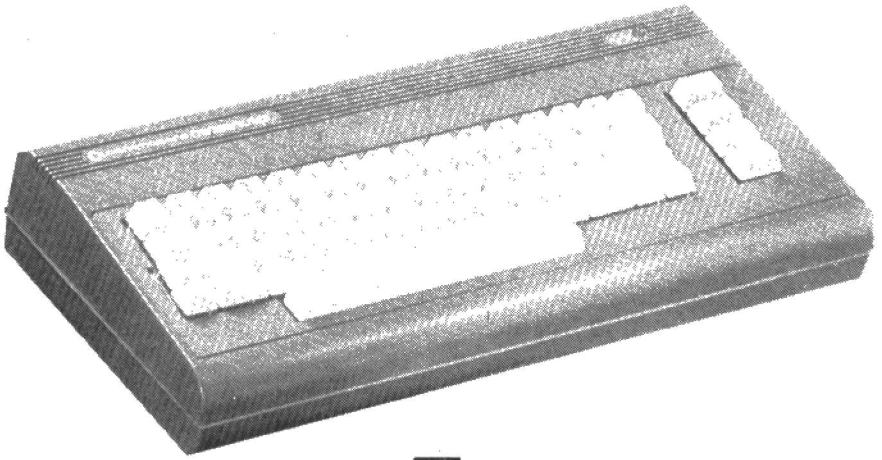
How about printing a few articles explaining the I/O ports of the smaller machines for the novice interested in connecting his machine to the outside world. As you pointed out in April's news section, it is now possible to buy an excellent control system for £44.00.

Stuart Watkins
Falkland Islands

Yes it is overkill, but a) a lot of our readers own BBC's, and b) who uses their computer while in bed asleep at night?

We have published articles on the I/O capabilities of the Oric and BBC micro in the past (and in this issue) and will be looking at others in the future.

A sneak preview of two new additions to the Commodore stable.



16 plus 4

At a recent dealer conference, Commodore put an end to the speculation surrounding their marketing plans for the rest of this year. The Company are to launch two new microcomputers that will bracket the established '64 both in terms of price and performance. The two new models are designated the C16 and the Plus/4, the C16 being the low cost replacement for the VIC20 while the Plus/4 is one of an increasing band of machines that aim to appeal both to the upper end of the home market and the businessman who wishes to assess the potential of a micro system at a relatively low cost.

The C16 is to be marketed as part of a starter pack that will retail at £129.99 and will include the computer, a cassette data recorder, an extensive guide to BASIC programming and four pieces of software. At this price the C16 seems a very attractive proposition for first time buyers and, although the 16K memory of the machine may seem meagre by todays standards, this is not likely to be too much of a disadvantage in this area of the market. We'll return for a closer look at the machine in the future but in the meantime our 'at a glance' panel gives the basic specification of the C16.

The specification of the Plus/4 reveals a feature which is likely to be adopted by an increasing number of machines, this being the inclusion of a number of application packages as part of a system's firmware. In the case of the 64K Plus/4 these consist of a word processor, a spread sheet program, a data base and a graphics package. All of these are available via a single keystroke and in addition to being able to share a common set of data, two of the packages can be viewed simultaneously. The Plus/4 is to retail at £249 but the £44.95 cost of the 1331 cassette recorder must be added to this in order to make full use of the machine. Once again, refer to the 'at a glance' box for a brief guide to the Plus/4's specification.

To complement the two new computers Commodore have launched a series of support peripherals. These are the MCS801 colour dot matrix printer that supports a colour, hi-res screen dump, the DPS1101 bi-directional printer; both of these are priced at £399.99. The 1342 single disk drive, at £229.00, is designed for use with both computers but like the '64 disk system is rather slow in operation. To overcome this problem, Commodore have designed the SFS481 fast disk system which adopts a parallel data transfer system. The price of this drive is yet to be announced.

> **"the Plus/4 includes a number of application packages as firmware"**

The 1703 colour monitor at £229.00 completes the range of new peripherals.

Hands on

All of the new hardware was shown at the recent Commodore computer show in London. The new models will be available in quantity in time for the run up to the Christmas period and the spread of specifications and price means that Commodore are well represented in all sections of the home micro market.

C16 and Plus/4 at a glance

Memory
C16 16K (12K available user programs)
Plus/4 64K (60K available for user programs).

ROM
32K ROM Standard (includes operating system and BASIC interpreter).

Microprocessor
7501 Microprocessor.
.89 to 1.76MHz clock.

Display
40 columns x 25 lines of text.

Colours
121 colours (15 colours:
8 luminance levels + black)

Characters
Upper and lower case letters, numerals and symbols.
Reverse and flashing characters.
All PET graphic characters.

Display Modes
Text characters.
High resolution graphics/multi-colour graphics.
Split screen text/high resolution graphics or multi-colour.

Resolution
320 x 200 Pixels.

2 Tone generators or
1 Tone and 1 white noise generator.

Tone
9 Volume levels.

Keyboard
Full size/full stroke design.

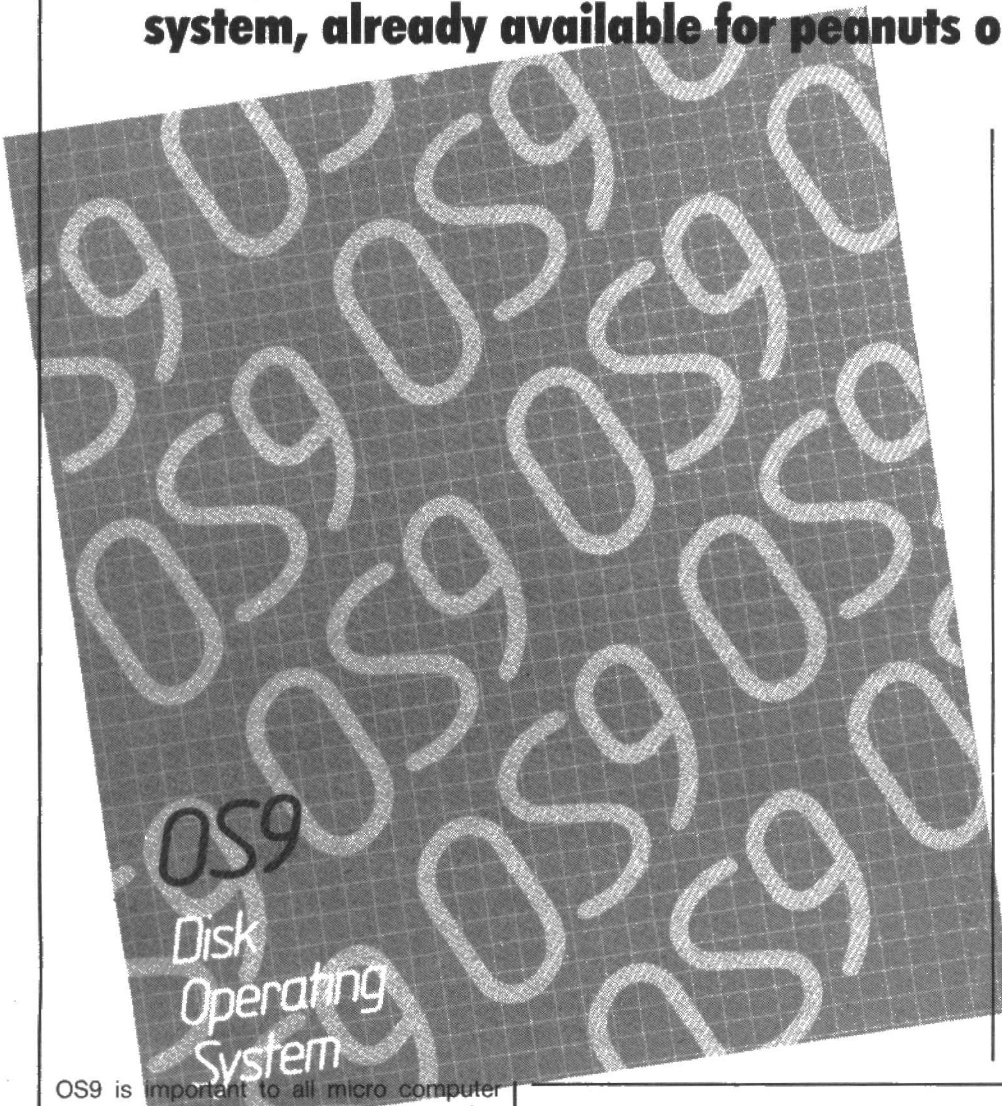
Keys
67 keys total.
4 cursor control keys.
4 programmed (reprogrammable) function keys (up to 8 user defined functions possible).
Colour control keys.
HELD key.
Upper and lower case character set.
Graphics character set.
Reset button*.
Escape key*.

Inputs/Outputs
User port.
Commodore serial port.
ROM cartridge and parallel disk drive port.
2 joystick ports.
C1531 Cassette unit interface port.
Monitor output – composit/chrominance/luminance.
Audio input/output.
Power supply input.

Features
Built-in BASIC 3.5 – over 75 commands including built-in graphics and sound commands.
Built-in Machine Language Monitor with 12 commands.
Screen Window capability.
*Not C16.

OS9: multi-tasking for the common man

UNIX is the word on the lips of the computer world, but what of OS9? Mike James raves about this multi-tasking operating system, already available for peanuts on the Dragon 64.



OS9 is important to all micro computer users because it is the first real operating system to be offered for a price that home users can afford. OS9 on the Dragon 64 is only £39.95 (including VAT) and this makes it one of the cheapest operating systems irrespective of quality or performance! The reason for my enthusiasm is that at the 'upper', ie expensive, end of the computer market there is currently a lot of talk and excitement about big operating systems that look and behave like UNIX.

UNIX mania has been around for about a year now and it shows no signs of growing weaker, but until OS9 there really was no way that anyone with a low (roughly less than £5,000) budget could get involved. OS9 is a UNIX 'look alike' that is itself cheap AND runs on low cost hardware. This means that for the first time it is possible for the real computer enthusiasts to get involved and both share in the excitement of something new and show the high cost computer users what it's all about.

Just think what computing would be like if CP/M has been sold originally for the same price as UNIX is today – a lot of CP/M software would never have been written – that's for sure! I'm not suggesting that OS9 will become the CP/M of the future but an operating system that incoporates many of the features offered by OS9 most certainly will. With all this talk of the future it is possible to miss an important fact about OS9 – it is available NOW and there is already enough applications software that runs under it to sink a battle ship.

CP/M considered

The eight-bit micros that we have all grown to know so well, mainly the Z80 and the 6502, are not really powerful enough to run full operating systems. In fact most of the chunks of software that are currently referred to as 'operating systems' are really nothing more than software disk interfaces. That is, the only real job that a program such as CP/M does is to organise disk storage into named files. Some versions of BASIC, Applesoft for example, have even included the disc handling commands and so removed the need for a distinctly separate operating system. The jobs that a comprehensive system has to perform certainly include looking after disc storage, but this is only a special case of the more general activity of 'managing' the machine's 'resources'.

'OS9 is the first real operating system to be offered at a price home users can afford'.

The fact that a computer has a range of resources is more obvious when you look at a large mainframe machine. However, even with a small micro it soon becomes clear that different applications make different demands of the hardware. For example, one program might use very little memory but quite a lot of processor time, another might use very little processor time because it spends most of its time waiting

for data to be read off disk. Once you start thinking about it, it is obvious that every computer has a range of resources which are used more or less depending on the application in hand. These resources include:. processor time (more usually referred to as CPU time), memory allocation, disk space, printer use, etc. The purpose of an operating system is to manage these resources so that the users get best value from their machines.

Task switching

The way that we currently use microcomputers is interesting because it makes no attempt to manage or avoid wasting resources. For example, if a program isn't using the printer, it just sits there, if a program cannot use all of the memory that is available then it remains unused and so on. The key to making better use of a machine's resources is to get it to run more than one thing at a time. If there is any left-over memory then it seems sensible to load another program into it; if the printer isn't being used then why not find it something to print. Of course the snag with this excellent suggestion is that each program that you want to run will need its own processor, because one processor can only do one thing at a time. The way around this is to arrange things so that the processor is the main resource that is shared by a number of tasks. At any one moment the processor can only be doing one thing but this doesn't stop it from switching its "attention" between a number of different tasks. This task switching is usually arranged to occur so quickly that, to a user, it really does look as though the machine is doing more than one thing at a time.

'Task switching' is one of the main characteristics of an operating system but to make it possible there are a whole host of other things that have to be done. For example, if more than one program is going to be run at a time then the operating system must be able to manage the memory in such a way that more programs can be loaded at the same time. If you think this sounds like an easy problem then you haven't thought about such things as making sure that there is enough space to load a program, making sure that one program cannot use another's allocation of memory and packing as many programs in as possible. In the same way, the disc filing system has to be set up so that more than one program can use it at once and the same is true of the printer or any other peripheral connected to the system.

Finally, it is worth pointing out that there are two distinct ways of running more than one program at a time. You can allow a single user to do this, a process known as 'multi-tasking', or you can permit more than one person to use the system at the same time, this is called 'multi-user'. Up to this time most microcomputer operating systems have been 'single-user/single-tasking' systems. Now operating systems such as Unix and OS9 are threatening to bring us out of the dark ages into a 'multi-user/multi-tasking' world. As far as most personal computer users are concerned the most important part of this change is multi-tasking. It's not difficult to see that there is almost something contradictory about a multi-user personal computer!

OS9 multi-tasking

OS9 multi-tasking is very easy to use. When you first start OS9 running (by simply inserting the OS9 disk and typing BOOT) you are greeted by a log on message that asks you to enter the date. After this you are presented with the prompt "OS9!" which is an indication that you can type commands to load programs, catalogue disks etc. In fact this prompt is issued, not by OS9, but by a program running under it called 'the shell'. The shell is really nothing more than a command line interpreter that continually looks at the input from the keyboard and then makes the appropriate calls to OS9 to implement whatever commands it detects. The advantage of this is that you can specify a different program to be run in place of the shell when you first boot up. In this way it is possible to create applications disks that keep the user well away from OS9 by loading the text processing/accounts/stock control or whatever is needed as soon as the system is started.

If you type a command such as:

```
DIR
```

to the shell it immediately loads a program called CMD/DIR and sets it running as an independent 'task'. That is, the CPU will switch its attention between the DIR program and the shell in the manner described earlier and give the impression that both programs are being run at the same time. The DIR program reads the disk and prints a catalogue of file names on the screen (ie it gives the user a DIRectory of the disk). After loading the DIR program the shell simply waits for it to finish. Thus, even though the two programs are being treated as separate tasks, because the shell waits for the DIR program it looks as though only the DIR program is running! To stop the shell from waiting for the DIR program to finish all you have to do is add an ampersand to the end of the command:

```
DIR &
```

This will set the DIR program running and the shell will carry on reading input lines and acting on them. The only trouble with this is that both the DIR program and the shell are trying to send output to the screen and this causes a rather mixed-up display. Obviously if more than one program is going to be running at the same time there has to be some way of easily redirecting output and this is exactly what OS9 provides in its redirection operators "<" and ">".

Notice that OS9, (following Unix) treats all programs that you want to run as independent tasks, allocates them an area of memory and a share of the processor time. In this way multi-tasking is the same and if programs are to be run one at a time then each one has to wait for is predecessor to finish. This approach greatly simplifies the design of the operating system and is much easier than treating sequential execution as the norm and then trying to add on multi-tasking at a later stage.

OS9 is available on the Dragon 64

Technically the details of the way OS9 deals with multi-tasking are also interesting. All OS9 assembly language programs have to be written in 'position independent code' (usually abbreviated to PIC) that can be loaded and run anywhere in memory. Writing PIC is difficult if not impossible with the older eight-bit micros such as the Z80 and 6502 but the 6809 was designed to make it easy. All machine code and data is formatted into 'memory modules' that includes a header block that the system can use to find out what it is and large it is. Thus OS9's memory management problems are reduced to loading memory modules into free areas of memory. Memory is allocated in units of 256 byte pages and as long as there are enough free pages forming a continuous block to accommodate a module it can be loaded and run. Notice that, although it is quite feasible for there to be enough total memory free to run a module it is impossible because the free pages are scattered around memory and do not form a continuous block. This problem is known as 'memory fragmentation' and it is something that all multi-tasking operating systems have to cope with.

Apart from memory management OS9 also has to share the processor's time among the different programs loaded in memory. This is achieved by the use of a regular system interrupt every 100ms (ie 10 times a second) that causes the processor to 'consider' changing to running another program. The processor only considers changing because there may not be another program worth running! For example, if the only other program in

memory is waiting for a printer to be ready to accept another character of data (remember compared to computers, printers are very slow) there is no point in changing. In general programs in memory are either active and ready to run; waiting for the completion of another program; or sleeping, that is waiting for a specific period or for a specific signal. Obviously only the programs that are active are considered as possible candidates for running. Which of the active programs is selected depends on the user-assigned priorities and the amount of time that a program has been waiting.

One of the problems with multi-tasking operating systems is that they attempt to manage the computer's resources efficiently but they often take so much memory and processor time for themselves that they leave very little in the way of resources for user programs! This is a factor which discourages people from using Unix as it takes around 100K of memory just to load it! OS9 is remarkably different and takes less than 16K of memory (most of which can be in the form of ROM if necessary). By comparison single-user/single-tasking operating systems take around 10K and look distinctly greedy when you consider the services that they offer. All things considered OS9 is very small and efficient for an operating system of its type – this has no doubt something to do with the 6809 and the way it has been used.

'OS9 treats all programs as independent tasks, sharing memory and processor time'.

The filing system

If more than one program and possibly more than one user, is going to be handled by a machine then the disk filing system has to be more sophisticated than those offered by CP/M, Flex etc. Even in single-user systems the simple single catalogue approach to organising files can lead to difficulties. How often have you catalogued a disk in an effort to find a file only to be faced with pages of output listing hundreds of files that you have never heard of. The problem is particularly acute when a system is using hard disk drives or any other large capacity storage device. OS9 solves the problem of trying to find a file in a huge single directory by allowing the user to create as many directories as required. At first this idea of multiple directories sounds complicated but, in fact, it is a great simplification. In the OS9 filing system there are three types of file – data files, directories and devices. Data files correspond to the usual sort of file encountered in other disk systems – ie it can be used to hold text, data or programs. Directory files are, as their name suggests, files that contain the names and other information on other files in the system. So, for example, on the OS9 system disc there is a directory file called CMDS and one called SYS. The command:

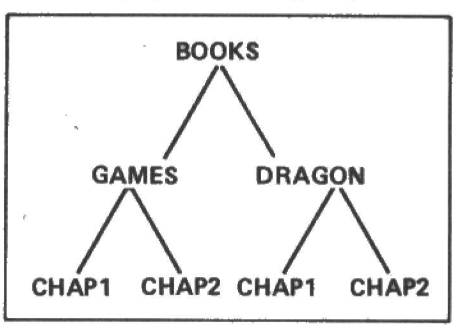
```
DIR CMDS
```

will give a listing of all the files in the CMDS directory and

```
DIR SYS
```

will give a similar listing for the SYS directory. The existence of a number of directories simply makes it possible to store files in appropriate groups; for example, all the system programs would be stored in the SYS directory. However, the real power of this method only becomes clear when you realise that there is nothing to stop any file being a directory. For example, the file TESTS in the SYS directory might itself be a directory, possibly containing the names of machine diagnostic programs. To find out what is stored in TESTS you would have to give its full name – DIR SYS/TESTS – ie TESTS is a directory within the SYS directory. To refer to a file in the TESTS directory, DCHECK say, you would have to use SYS/TESTS/DCHECK – ie DCHECK is the program stored in the TESTS directory, which itself is stored in the SYS directory. A file name that involves a number of directories in this way is referred to as a 'path name' because it is the route that you have to take through the directories to find the file.

The use of path names and multiple directories results in a file system that is often referred to as 'hierarchical' or 'tree structured'. For example, if I create a directory called BOOKS and within it I then create a two directories called GAMES and DRAGON and then within each of these I might create text files called CHAP1, CHAP2 and so on. This situation can be summed up by the following diagram:



You can now see that for example, the path name BOOKS/DRAGON/CHAP1 really does describe a 'path' through the filing system that leads to the file in question. Notice also that there can be more than one file with the same file name eg CHAP1, but each file always has a unique path name.

This sort of hierarchical filing system is easier to use than the single directory system. For example, if I wanted to discover what books were stored on the disc all I would have to type is DIR BOOKS. However, if the current book that I was working on was DRAGON, it would soon become tedious to have to type the full path name each time I wanted Chapter Two say – ie LIST BOOKS/DRAGON/CHAP2 when a listing was required. OS9 solves this problem by allowing each user to set up 'working directories'. For example, following the command

CHD BOOKS/DRAGON

a simple reference to CHAP2 would be taken to mean

BOOKS/DRAGON/CHAP2

```
                         System Device Directory
              +------------+-----------+---------+
              !            !           !         !
              D0          TERM         P         D1
              !                                  !
              !                                  !
              !                                  !
       D0 Root Directory               D1 Root Directory
  +-----------+-----------+         +-----------+-----------+
  !           !           !         !           !           !
 DEFS      startup      CMDS      file1       file2       file3
  !                       !
  !                       !
  !                       !
--+--         +-----+-----+-----+-----+
  !           !     !     !     !     !
OS9Defs     copy  list   dir   del  backup
```

(CHD stands for CHange Data directory), In this way you can avoid having to type full path names more than is strictly required. OS9 actually allows the user two different current directories – an execution directory where any programs that are needed are assumed to be stored and a data directory where data files are stored.

Another interesting feature of the OS9 filing system is the way that the existence of different storage devices is treated. Each device is associated with a file that describes it. These device files are set up at the time that the system is created and cannot be changed. For example, the main disk is usually referred to as D0 and a second disk would be D1. Device files can only occur at the start of a path name and they must be preceded by a slash. Thus DIR /D1 would give a directory of the files at the start of the hierarchy on disk one. This use of device files makes it possible almost to treat different devices as offshoots of the filing system tree.

There are various other details of the OS9 filing system that make it very easy to use but its essential characteristics are determined by the fact that it is hierarchical.

'The symbol "!" creates a channel of communication called a pipe between a pair of programs'.

Now that the purpose of an operating system has been outlined the time has come to look at OS9 and see how it compares to the theory. This month the multitasking abilities of OS9 are examined and the way that it handles discs and other I/O devices is explored. The second part of the article will describe how OS9 and the Dragon get along together and will provide an overview of the vast range of software that runs under it.

Other OS9 I/O

The main part of OS9's I/O system has already been described in the last section. The filing system is a very important part of any operating system but for the user it is just one of a number of places that data can be sent to and received from. For example, you can send data to a disc file or you can send it to a printer and in an ideal operating system there should be no real difference between the two. In other words you should be able to write a program that generates data without worrying about where it is going to. In OS9 all I/O is treated as a stream of bytes and different I/O devices are treated as device files. For example, the printer is associated with a device file called 'p' and so it is as easy for a program to send data to the printer as to a disk file. This similarity between all types of I/O can best be seen in the way that the shell can be used to redirect I/O. Normally the DIR command sends its output to the screen but this can easily be changed:

DIR BOOKS>CAT

will send its output to a disk filed called CAT (in the current working directory) and

DIR BOOKS>/P

will send the output to the printer. In the same way the '<' sign can be used to make a program take input from somewhere else other than the keyboard. For example,

UPDATE <BATCH1

will make the program UPDATE take its instructions from a disk file called BATCH1 instead of the keyboard. You can use both '<' and '>' to redirect both input and output to a program to a disc file or any other device.

I/O redirection is of most use when you are trying to run a number of programs at the same time and don't want them all to be trying to use the screen and the keyboard at the same time. However there is another sort of I/O redirection that allows communication between a pair of programs that are running concurrently. For example, if you use the command line:

UPDATE<MASTER–FILE ! SORT ! FORMAT–TABLE >/P

then the three programs UPDATE, SORT and FORMAT–TABLE will be loaded and run concurrently. The input to the UPDATE program is taken from the disk file MASTER–FILE and its output is sent, as a stream of bytes to SORT, which then passes its outputs as FORMAT–TABLE, finally the output of FORMAT–TABLE is printed. The symbol "!" creates a channel of communication called a 'pipe' between a pair of programs that are being executed at the same time. OS9 automatically looks after the transfer of data between the program and makes sure that the program that is waiting for input never 'gets ahead' of the one that is generating the output.

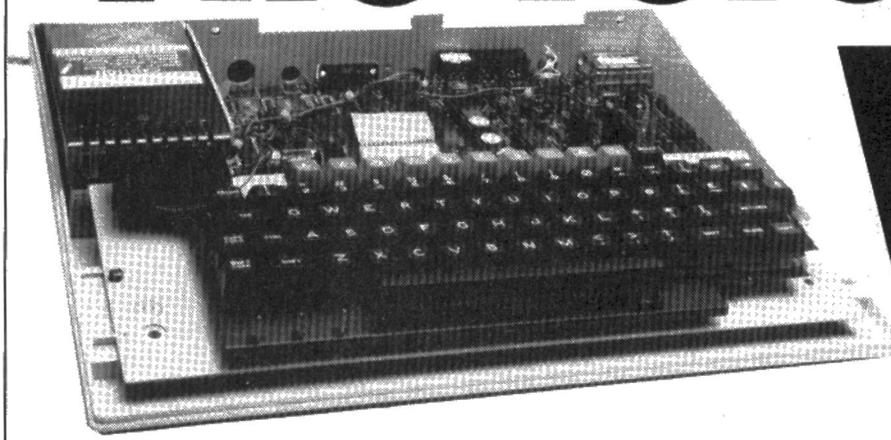
The unified system of I/O is one of the many advantages of using an advanced operating system. It is all to easy, however, to concentrate on individual details of an operating system and not see how they fit together. For example, combining I/O redirection with the multi-tasking capabilities of OS9 gives what other operating systems refer to as 'printer spooling'. For example:

DIR>/P&

sends the output of the DIR command to the printer as a separate task. In other words the shell returns immediately, ready to accept other commands from the user while the printer is kept busy printing the directory – this is what resource management is all about!

In this all too brief look at OS9 its advantages and methods of working have been described but without reference to any particular piece of hardware. Obviously an operating system ease of use is also a function of the power and quality of the hardware that is running it. Next month the subject of how well the Dragon 64 runs OS9 and how suited they are to each other is taken up along with a brief survey of some OS9 software and the incredible BASIC 09 which is certainly a better BASIC.

The Versatile VIA



Paul Beverley, author of the new Service Manual for the BBC Microcomputer, continues this series of articles in which he tries to explain some of the complexities of the 6522 versatile interface adaptor.

Having looked last month at the first main feature of the versatile interface adaptor and its two 8-bit parallel ports, we can now turn our attention to the counter-timers. The VIA contains two 16-bit binary down-counters which can be used either to count pulses from outside the computer, or to count the clock pulses applied to the chip from the internal system clock (1MHz on the BBC microcomputer despite having a 2MHz clock for the 6502 itself). Counting 1MHz clock pulses also provides a timing facility since the value held in the counter-timer registers represents a time in micro-seconds. This is why they are referred to as both timers and counters, but rather than talking about "counter-timers" all the time, I shall use the terms "counter" and "timer" interchangably.

The description of the chip as a whole as being "versatile" applies equally to the timers, but because of their versatility they are also rather complex. They can be used to generate time delays, to measure time or frequency and to act as pulse generators. In order to understand how they work, we will look at them first of all in general terms, and then try to show a variety of applications to illustrate the principles.

In referring to the various addresses associated with the timers I shall be using those of the "external" VIA of the BBC microcomputer – the one used for the User Port and Printer Port. This has the address range, &FE60 to &FE6F. If you have a different microcomputer that also uses the 6522 VIA, then you will need to change the base address, which is the first three hexadecimal digits. There are seven addresses associated with the timers: six addresses used to transfer data to and from the timers (&FE64 to &FE69), and one, the auxilliary control register (&FE6B), which is used to control the modes of operation of the timers. The other two addresses which we will need to use later are &FE6D and &FE6E, the interrupt flag register and the interrupt enable register.

There are actually seven registers associated with the six data transfer addresses, and there is by no means a one to one relationship between addresses and registers. The two timers referred to as T1 and T2 are represented in **Figure 1** as four 8-bit registers using addresses &FE64, &FE65, &FE68 and &FE69. These are the addresses used to read the values in the two 16-bit timers.

Timer 1 is connected permanently to the 1MHz clock pulses, and is therefore continuously decrementing. The basic use of the timers is to produce time delays by loading a particular value into them and letting them count down towards zero. When they reach zero, the processor can be informed of the fact either by the VIA setting a flag, ie setting one bit of the interrupt flag register to logic 1, or by generating an interrupt.

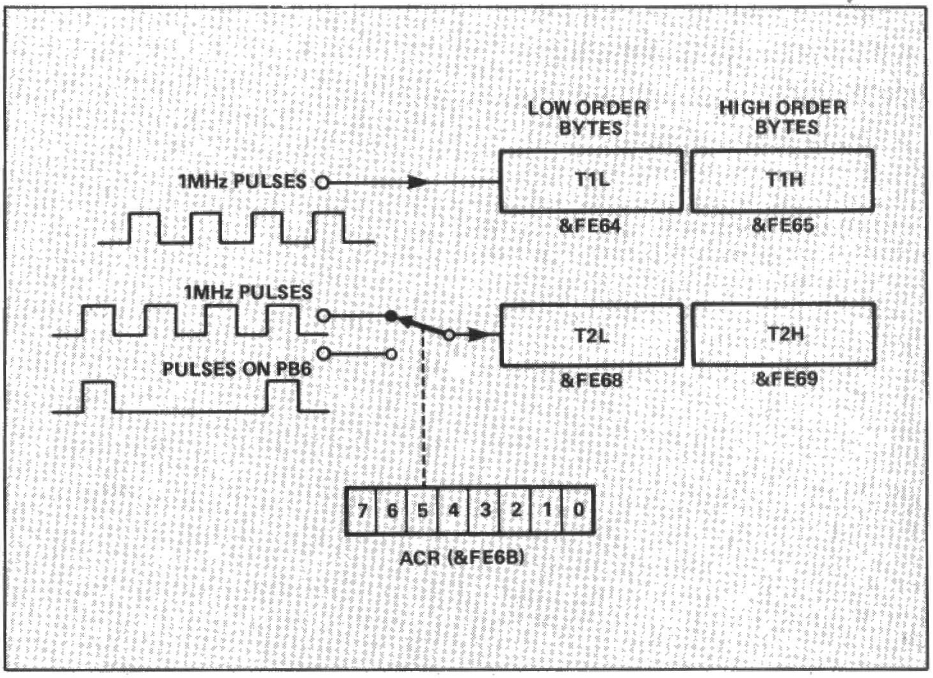


Figure 1. Basic addresses of the two timers and possible pulse sources.

As shown in **Figure 1,** Timer 2 has two possible sources of pulses. The first is the same as Timer 1 – the 1MHz clock pulses – but the alternative is PB6 which can be

used to take in a stream of pulses from some external source. Switching between the two is achieved, not by a hardware switch, but by changing the contents of bit 5 of the auxilliary control register.

One consequence of the fact that Timer 1 is permanently fed from the 1MHz pulses is that it is continuously changing. To illustrate this, using a BBC micro, try typing in:

```
10 REPEAT
20 PRINT ~!&FE64 AND &FFFF
30 UNTIL FALSE
```

This picks off the values of the low and high bytes of Timer 1 and displays them as a single hexadecimal number. The number will be seen to be changing continuously. Better still, if you have a Disc Doctor ROM, try typing in *MZAP FE60 and you will see a continuous display of the contents of all the registers of both the internal VIA and the external VIA and the external VIA (two copies of each in fact. Since the address decoding is not complete, ie the internal VIA can be accessed either from &FE40 to &FE4F or from &FE50 to &FE5F and similarly the external VIA can be accessed either from &FE60 to &FE6F or from &FE70 to &FE7F). The contents of all the four addresses given in **Figure 1** will be seen to be changing, as will those of Timer 1 in the internal VIA (&FE44 and &FE45), whereas Timer 2 in the internal VIA (&FE48 and &FE49) is not counting. The reason is that

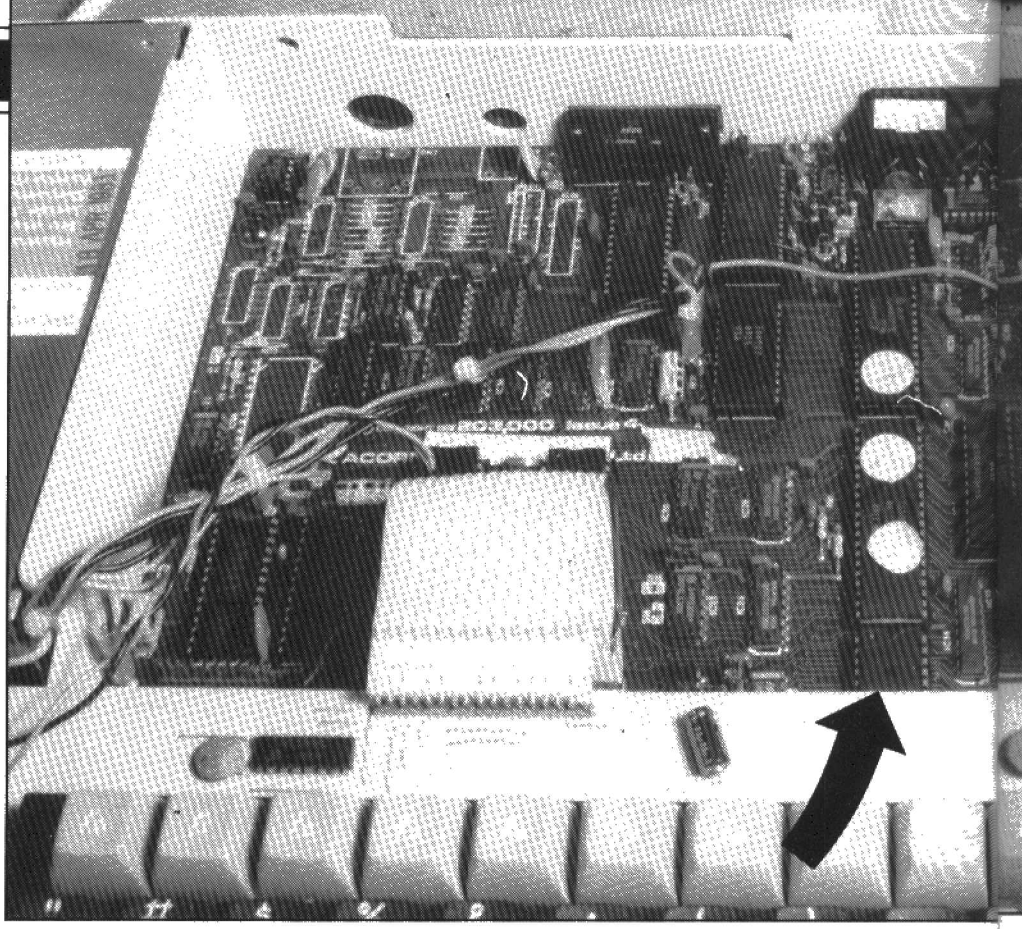
'timers can be used to generate time delays, measure time, or act as pulse generators.

Timer 2 is set up by the operating system ready to count pulses coming in on its PB6 line from the speech chip, but since there are no pulses, its value is not changing.

Register writing

Having established that it is not too difficult to read the timers using four addresses for the four registers, we turn our attention to writing to these registers. The problem is that we want to load a 16-bit counter, but the 6502, having an 8-bit data bus, can only handle 8 bits of data at a time. This means that if you load one of the 8-bit registers then by the time you have loaded the other byte, the first byte may well have changed. To overcome this problem, the designers of the VIA have provided extra registers which are referred to as "latches". The idea is that each latch is a temporary store for the number which is to be transferred to the counter register when required. Timer 2 having only one latch, illustrates this more simply than Timer 1 which has two latches, so we will look at Timer 2 first.

Figure 2 illustrates what happens when reading and writing the addresses associated with Timer 2. As we have already said, reading these two addresses simply gives the current values of the



The BBC 6522 VIA chip (arrowed).

upper and lower bytes of Timer 2. However, if you write into &FE68, the data is not transferred into the counter register directly, but into the latch. It is only when you write into the high byte of the counter, using &FE69, that the VIA triggers the transfer of the data from the latch into the low byte of the counter register. This means that the action of writing to the high byte effectively loads all sixteen bits at once, and the timing or counting then starts.

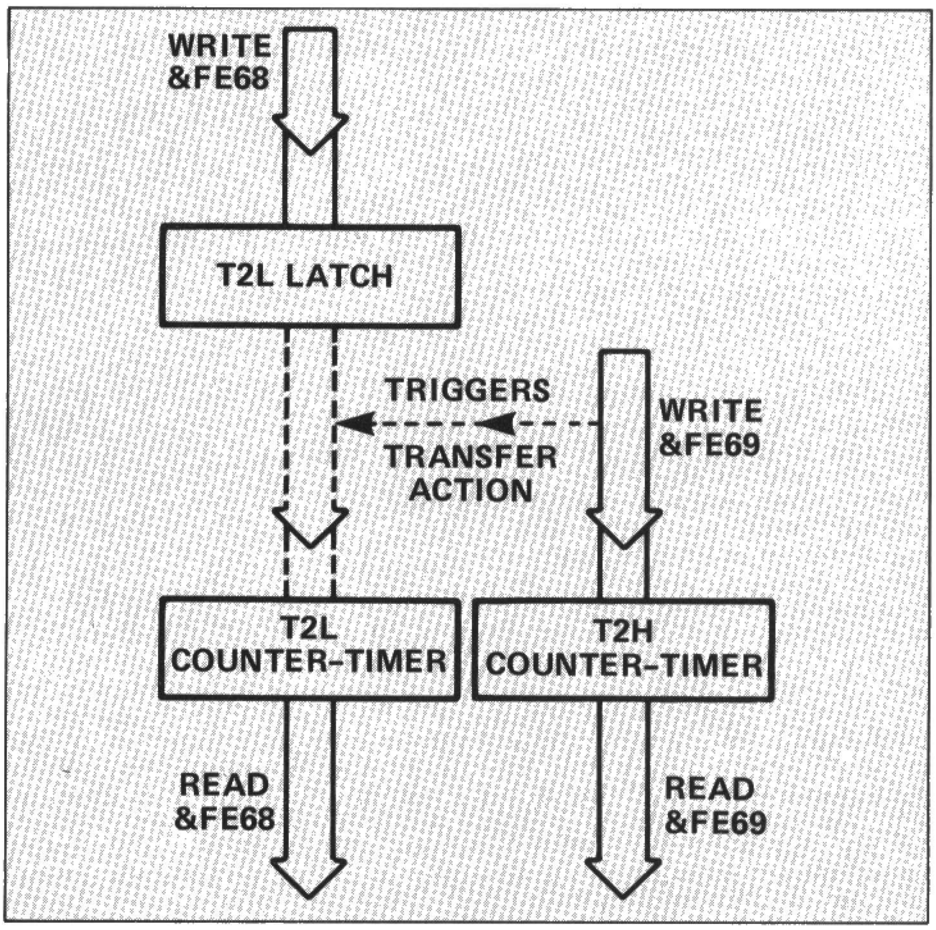
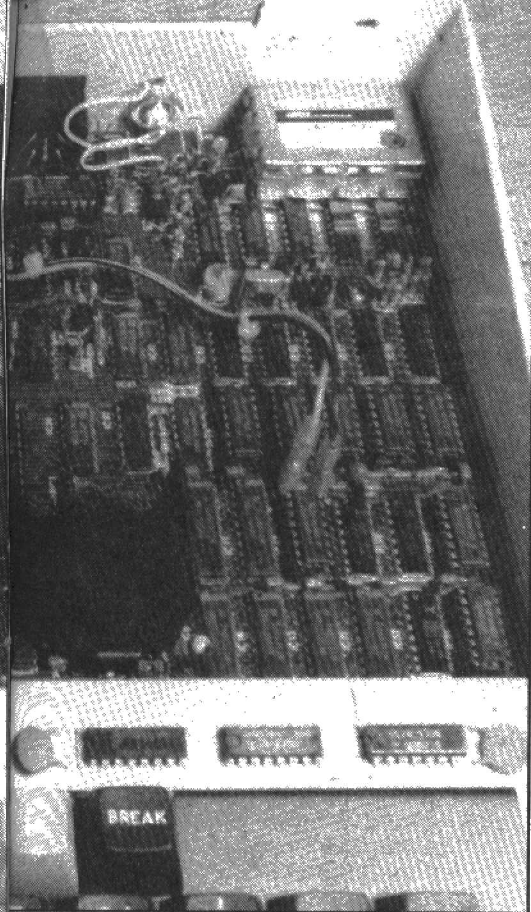


Figure 2. Data transfers for timer 2.



Timer 1, as shown in **Figure 3,** has four registers and four addresses. Reading the four addresses gives you the values contained in the counter registers and the latches respectively. However if you try to write to the low order counter (&FE64), then the data does not go into the counter but

"The practical examples use BASIC, there is no need to resort to machine code".

straight into the latch. In other words, it has exactly the same effect as writing directly into the latch at &FE66. If you write to register &FE67 then the data only goes into the high order latch, but it is not transferred into the high order counter.

Writing to register &FE65 puts the information into both the latch and the counter register. Also, as with Timer 2, writing to the high byte of the counter has the effect of triggering the transfer of the data from the low order latch into the low order counter. Thus we have effectively a sixteen bit load occurring when we write to &FE65.

The extra latch associated with Timer 1 is not necessary for normal loading of the timer, but is used in the so-called "free-running" mode, as opposed to the usual

However, although the timer continues to decrement indefinitely, no further signalling to the processor occurs unless the timer is re-loaded and it "times out" again. The one-shot mode also applies to Timer 2 when it is counting pulses coming in on PB6. The original design intention was that the user specifies the number of pulses to be counter and this value is stored in Timer 2. When that number of pulses has arrived, the T2 interrupt flag is set and an interrupt can be generated if needed.

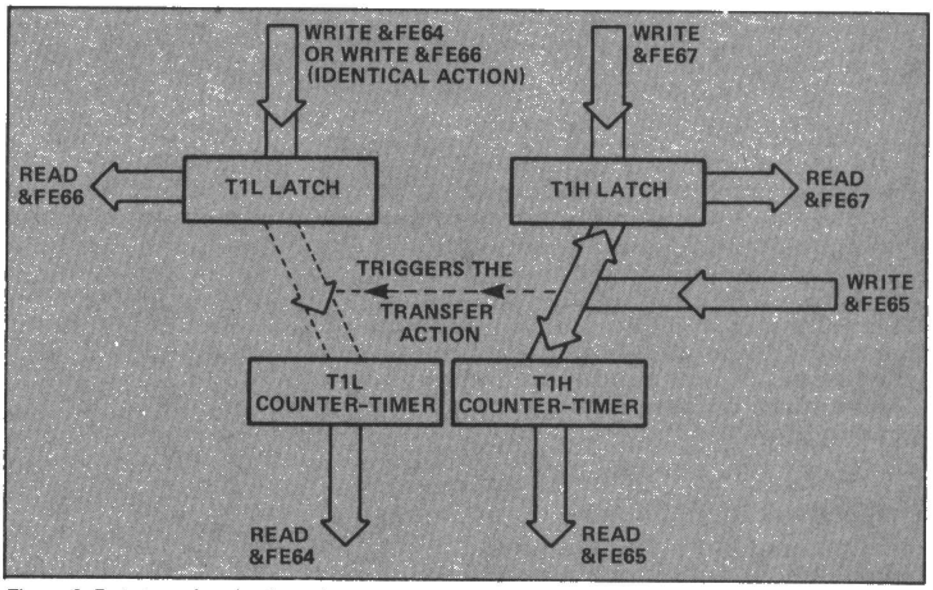


Figure 3. Data transfers for timer 1.

"one-shot" mode. In the free-running mode, every time the timer goes through zero, the VIA, as well as signalling the processor, also automatically re-loads the timer from the high and low order latches. This process will carry on ad-infinitum or until the timer is changed back into the one-shot mode. A typical application of the free-running mode is to produce a regular train of interrupts such as those used on the BBC microcomputer to up-date the centisecond clock for the TIME facility.

In the one-shot mode the number is loaded into the timer, and when this passes through zero the processor is signalled.

BIT NUMBER	LOGIC 0	LOGIC 1
7	PB7 input or output as controlled by bit 7 of data direction register	PB7 output from Timer 1
6	Timer 1 in one-shot mode	Timer 1 in free-running mode
5	Selects 1MHz clock pulses as input to Timer 2	Selects pulses from PB6 as input to Timer 2
2,3,4	Controls shift-register operation	
Ø,1	Controls latching of data on PA and PB	

Figure 4. Summary of the auxilliary control register functions.

Another useful facility which the VIA has, associated with the Timer 1, is the possibility of producing an output on PB7 to signal the end of the timing interval. In the free-running mode, every time the timer reaches zero, the state of the PB7 line is changed. In the one-shot mode, when Timer 1 high byte is loaded, PB7 goes from logic 1 to logic 0, and then at the end of the time interval, it returns to logic 1.

Control of the various functions of the timers is achieved by using three of the bits of the auxilliary control register (ACR) at &FE6B. Bit 5 relates to Timer 2, and bits 6 and 7 to Timer 1. A summary of these is given in **Figure 4.** Bit 5 selects either 1MHz pulses or pulses coming in on PB6, to be applied to Timer 2. Bit 6 selects either the one-shot mode or the free-running mode for Timer 1, and bit 7 is used to determine whether PB7 is an output from Timer 1, or whether it is either an input or an output as controlled by bit 7 of the data direction register, as explained last month. A number of the published 6522 data sheets are incorrect at this point since they say that bit 7 of the data direction register must be set to logic 1 as well as bit 7 of the ACR in order to get PB7 to act as an output for Timer 1. This is not so – only the ACR, but 7 must be set, and when it is set, reading &FE60 (port B) shows the current value of the PB7 line as set by the timer. Whether or not bit 7 of the data direction register is set, if bit 7 of the ACR is set, writing to &FE60 has no effect whatsoever on PB7 – it is entirely under the control of Timer 1.

One interesting but possibly useless fact

LISTING 1 – To produce a square wave output on PB7.

```
10 ?&FE6B = &C0 : REM T1 free-running
20 ?&FE65 = 0   : REM Start the timer
30 L% = &FE66   : REM T1 low latch
40 H% = &FE67   : REM T1 high latch
50 FOR N% = 0 TO 65535 STEP 10
60   ?L% = N% MOD 256
70   ?H% = N% DIV 256
80   NEXT
90 END
```

LISTING 2 – To measure the time of a BASIC program line.

```
10 ?&FE6B = &20 : REM Timer 2 pulse counting mode
20 ?&FE68 = &FF : REM Start with maximum values
30 ?&FE69 = &FF
40 ?&FE6B = 0   : REM Switch to 1 MHz pulses
50 REM line to be timed
60 ?&FE6B = &20 : REM Stop Timer 2
70 PRINT &FFFF - (?&FE68 + 256 * ?&FE69) - 842
```

LISTING 3 – To give the average and minimum execution times of a BASIC program line.

```
10 X% = 1000 : REM number of timing
               samples for averaging
20 B% = 0
30 E% = &20
40 A% = &FE6B
50 ?A% = E%   : REM switch 1 MHz pulses off
60 MIN% = &FFFF
70 FOR N% = 1 TO X%
80   ?&FE68 = &FF
90   ?&FE69 = &FF
100  ?A% = B%
110  REM Line to be timed
120  ?A% = E%
130  T% = &FFFF - (?&FE68 + 256 * ?&FE69)
140  IF T% < MIN% THEN MIN% = T%
150  total% = total% + T%
160  NEXT
170 T% = total% / X%
180 PRINT "Average value "; T% - 536
190 PRINT "Minimum value "; MIN% - 513
```

is that if PB6 is set as an output by putting a 1 in bit 6 of the data direction register B (&FE62), then if you write 1's and 0's to PB6 using address &FE60, it will actually cause Timer 2 to decrement just as if the pulses were coming in from outside the VIA. This might be useful if you wanted to send out a specified number of pulses, counting them automatically on T2.

Flags and Interrupts

We have shown then that PB7 can be used to signal devices outside the computer that the end of a time interval has been reached. However, in order to alert the processor itself, the interrupt flag register at &FE6D is used. This has two bits which are used as flags to show whether each of the timers has "timed out", as the data sheets put it. If bit 5 is set, then Timer 2 has timed out and if bit 6 is set then Timer 1 has timed out. The processor can therefore determine whether either timer has reached zero by checking the appropriate bit. The flags can then be re-set to zero either by reading the appropriate low order timer register, or writing the high order timer register.

The alternative approach is to get the VIA to generate an interrupt automatically when either of the flags is set. If one of these interrupts is enabled, then as soon as the flag is set, the VIA pulls the IRQ line low. This signals the processor to go off into its interrupt service routine which has then to include a check to see whether the VIA itself is the source of the interrupt, and if so to respond accordingly. This will be dealt with in more detail in a later article.

Applications

To illustrate practical examples of the use of these timers, I have chosen two applications which can be implemented in BASIC so that you do not need a knowledge of machine code, and also so that they are easy to translate onto other microcomputers. The first program uses Timer 1 in the free-running mode with PB7 as output (ACR =?&FE6B = &C0 = 1100 0000 binary = 192 decimal) in order to generate a square wave pulse. The listing given as program 1 starts by generating the highest possible frequency (timer value = 0) which produces a frequency of 250kHz and then sweeping through to the lowest output frequency which is 7.6Hz (131 milliseconds). The PB7 line inverts once every time the timer counts down through &0000 to &FFFF and then takes 1 microsecond to reload it, thus N% is the number loaded into the latches, the time between the inversions is N% + 2, and thus the period of the output pulses is 2 * (N% + 2) microseconds. Therefore N% = 0 produces a pulse with a period of 2 * (0 + 2) = 4 micro seconds ie a frequency of 250kHz.

Once the pulses have been started by writing to &FE65, the program only alters the contents of the latches (&FE66 and &FE67) rather than the timers. The reason is that if you change the timers, the pulse re-starts as soon as the high byte is loaded rather than finishing the current timing cycle. This tends to produce shortened pulses which have a larger proportion of higher harmonics, which at audio frequencies is quite noticeable.

Another effect which you may notice with this program is that, because the two bytes of the new number are loaded separately at the relatively slow speed of BASIC, it produces a slight hiccup each time the high byte of N% changes. To overcome this, on the BBC micro, you can use !L% = N% in place of lines 60 and 70. The only trouble with this is that !L% is a four byte "poke" so that it would poke zeros into &FE68 and &FE69, and so Timer 2 could not be used in the same program.

If you have access to an oscilloscope you can look at the square wave produced on PB7. Alternatively if you have an audio amplifier you could "listen" to the pulses, although you won't hear much at 250kHz! Remember though that the pulses are 0V to +5V and it is therefore a good idea to feed them into a 10k or 100k potentiometer and take the signal from the wiper of the potentiometer to the amplifier input with the 0 volts line to the screen or ground of the input.

Timing BASIC program lines

The second application is measuring the execution times of BASIC program lines. This is done by using the fact that Timer 2 can effectively be stopped and started by switching its pulse source between the 1MHz clock and PB6. If there are no pulses coming in, the selecting PB6 stops the count and the value that it has reached can be read even from within BASIC without getting incorrect readings. The simplest version of this is shown as **Listing 2.** If you run this on the BBC micro but do not have any line 50 at all, then you should get an answer of zero. This can be adjusted by changing the number subtracted in line 70. However, what you will find is that the number you get tends to vary – sometimes 0 and sometimes other values. The reason for this is that in the BBC microcomputer considerable use is made of interrupts for the normal running of the machine. Thus, if an interrupt occurs during the execution of a BASIC program line, the interpretation is halted temporarily whilst the VIA timer keeps ticking away. The overall execution time is therefore increased by the length of time taken to service the interrupt. On a microcomputer which does not make use of regular interrupts, the time value given by this program would not vary.

Listing 3 takes account of this variation which is due to interrupts, times the line under test a number of times as set by X%, and comes up with the average time taken and also the minimum (uninterrupted) time. Again, the numbers subtracted in lines 180 and 190 are such as to give zero for both the averaged and minimum times with line 110 deleted altogether. The values shown are those when running the program without any REM statements.

"The description "versatile" applies to the timers".

In next month's feature, we will look at the idea of using the timers to measure the times, in microseconds, of various events either occurring within the computer, or signalled from outside by changes on one or more of the port lines acting as input ■

Improve your view

Brian Alderwick and Peter Simpson have a few improvements for those of you who have bought Acorn's 'VIEW' wordprocessor.

Many of the people who have bought the Acorn 'VIEW' word processor for their BBC micros have found it to be a very valuable aid in the production of well laid out documentation. One large drawback that many people experience is the limited provision for sending commands to the printer. Modern printers, such as the Epson RX80 and FX80, have a large number of special features which it might be desirable to use, and the provision of only two highlight keys makes access difficult.

The normal method of interfacing the word processor to the printer is by the use of a printer driver. This program translates highlight codes from VIEW into appropriate control code sequences to send to the printer. The printer driver program is written in assembly language and assembled into machine code to live in page four ie &400 to &4FF. The first restriction that one comes up against when writing a driver is that 256 bytes is just not large enough, since it is difficult to squeeze more than about six different functions into one page.

The second restriction is that only two highlight codes are simultaneously available from the function keys as defined in VIEW. If more are needed the codes must be redefined with the 'HT' command whilst typing in the text.

A dump or hex listing of the contents of the VIEW ROM shows that only about 12K out of 16K is used which leaves more than enough room to provide extensive built-in driver routines. It is possible to transfer the program to a 27128 EPROM and add the additional code for the drivers. To effect the transfer, access to an EPROM programmer is necessary. An alternative is to use a 'sideways' RAM board.

BEEBUG's EXMON utility chip has a facility to copy any sideways ROM down into RAM. Furthermore, Solidisk Technology have been supplying purchasers of their RAM boards with programs to copy both the Acorn DFS 0.90 and the Computer Concepts Wordwise chips so that modifications may be made to the code to introduce new functions. Next month *E&CM* will feature its own listing to enable you to download the ROM onto EPROM and add the improvements in **Listing 1.**

Features

The accompanying programs add the following facilities to VIEW A1.4 without the need to load a printer driver from disc or cassette:-

1) VIEW enters a default screen mode which can be chosen at the time of programming the EPROM and will probably be MODE 3, 6 or 7.

2) VIEW will automatically engage a named built-in printer driver which can be chosen from EPSON (for the MX, RX and FX printers), JUKI (for 6100 model), or DEFAULT. Again, this choice is made before programming.

3) Ten additional highlights are provided. These are selected by depressing SHIFT/CONTROL and one of the red function keys as shown in **Table 1.** VIEW normally has two highlight commands (SHIFT f4 and SHIFT f5) and if one wishes to use more than these in a document, then one or other of the keys must have its definition changed with the HT edit command (SHIFT f8) followed by the new highlight code. The new software provides ten additional highlight keys making such redefinitions unnecessary. Highlights 1 and 2, SHIFT f4 and SHIFT f5, still give their normal default values of highlight code ie 128 and 129 respectively and appear as their normal characters, but in inverse video. Codes 130 to 139 are obtained by simultaneous depression fo SHIFT/CONTROL and one of the function keys f0 to f9. In other than MODE 7, highlight code 130 is shown as an inverse video up arrow (for superscript) and highlight code 131 is shown as an inverse video down arrow (for subscript). Highlight codes 132 to 139 are denoted by inverse video numbers 2 to 9 respectively. This gives a very clear indication of these codes at a glance.

Unfortunately, if MODE 7 is used, the characters representing these codes will be in normal video as it is not easy to achieve inverse video in this mode. In MODE 7 highlight codes 130 to 139 are represented by normal video numbers 0 to 9 respectively. Users who wish to use a 40 character display and have the enhanced highlight markers, may use MODE 6, but of course, this would leave less memory free than in MODE 7. The new highlight codes are used in exactly the same manner as the original two highlight codes. Code 138, SHIFT/CONTROL f8, gives one and a half line spacing and this was found useful. However, since it is not provided as standard in VIEW, no account of this is taken when VIEW counts the lines to determine the page length. Therefore, when using code 138 the page length should be set to two thirds of its original value with the edit command PL. Thus if the default PL of 66 is normally used when PL should be set to 44.

Any documenmts prepared on the standard VIEW will reflect these changes, that is, inverse video highlight markers will show in screen modes other than MODE 7. Any redefinitions of the HT command will act as normal. It should be noted that some of the above highlight codes cannot be used simultaneously, this being a function of the printer, full details are given in the relevant printer manual.

4) A new edit command, the CL command has been added to simplify entry of global highlight codes. This is obtained by entering the new edit command ie SHIFT f8 and then CL <return>) followed by a highlight name as in the table. That highlight will then become effective. It should be noted that two highlights or more can be given in the same command, but they must be separated by a comma or space. Where changes of highlight are needed in midline, the normal highlight system can still be used. If the highlight is to be turned off a second identical command should be given. Thus CL ELITE, LINEANDAHALF will select the elite typeface and line and a half spacing.

5) The facility in 4 above has been extended to enable any sequence of numbers to be sent directly to the printer. This allows functions not covered in the twelve provided to be activated and is identical to the Wordwise OC command. The numbers are sent by using the CL edit command followed by SENDPRINTER and the numbers required. These should be decimal numbers and separated by spaces or commas eg SHIFT f8 CL <return> SENDPRINTER 7 would sound the bell on an Epson printer. This system will sends commands to any printer, not only those for which drivers are inbuilt.

6) A pad character has been added which will be converted to a space when printed but which otherwise behaves as a printable character. This is useful when it is preferable that words should not be split over two lines or when it is desirable that soft spaces be prevented from being inserted at a particular place. The pad character is produced by simultaneously pressing SHIFT/CONTROL and the COPY key. It is represented in the text by an inverse video P.

7) A backspace facility has been added that will allow the printer head to move backwards. this can be useful for printing superscripts and subscripts, one above the other such as is often the case with equations. The key sequence which

produces the backspace is SHIFT/CONTROL and a simultaneous CURSOR LEFT. The character will be displayed as a inverse video back arrow. It should be noted that if a backspace is used to do dual super/subscript then the longer of the pair should be done last so as not to overwrite the shorter of the pair when continuing to print normally. The column counter has been modified to account for backspaces, therefore, formatting will not be affected. If MODE 7 is used for either of thew facilities in 6 or 7, the video will be normal and not inverse for the P and back arrow.

8) Both the £ sign and # are now available directly from the appropriate key for the built in drivers. It is not necessary to change the character set between English and USA to achieve this.

9) In VIEW the CONTROL/f9 user defined key has not been used so it has been utilised to provide a °C. This should be useful in academic or scientific circles.

10) One problem with using VIEW in the sheets mode is that the printer driver on and off commands are called at the beginning and the end of each sheet respectively. This can cause problems because the initialisation routines often clear the highlight flags. Thus it becomes necessary to switch the required highlights such as double-strike or elite on at the beginning of every page. The new software overcomes this problem by setting a flag in RAM at &100 to indicate whether the open or close is the first or last in the document. Intermediate on and off calls may thus be treated differently to the initial and final calls, circumventing the problem.

11) Another shortcoming of VIEW is the fact that if a highlight is set in a header then that highlight will be ignored. Code has been added which overcomes this problem and highlights will now be treated correctly.

12) Three characteristics have been found in VIEW that all affect the formatting of a document and have been changed such that:-

A) HT codes are ignored in determining how many words will fit on one line.

B) Soft Spaces are not converted to hard spaces, as has been found under certain editing conditions.

C) Lines are reformatted only when necessary when in the Insert mode.

13) The *HELP STORED command will now display all the highlight commands together with their codes and characters as they will appear in the current screen mode as well as details of the additional facilities and the status of the power up options.

Modifications

The modifications are effected by running the program in **Listing 1** once the ROM is copied to EPROM. **Listing 1** should be typed in omitting the leading spaces in the assembly language statements and the REMs to ensure that enough memory is available for the program to run.

After entering **Listing 1,** run it and answer 'Y' to the test run question. A default set of options will be selected automatically and the machine code will be assembled. Checksums will then be calculated for each 128 byte block and compared to the correct values. If an error is found, an appropriate message is printed describing the address range within which the error was located. Should an error be found then re-run the program and use CONTROL/SHIFT to stop the scrolling as necessary to examine the screen listing. From this, determine which assembly language instructions have been assembled into the address range and check these against the magazine listing. Remember that an error in one block may cause all subsequent blocks to be apparently incorrect, so the recommended procedure is to re-run the test option after each error is found and corrected. When the program runs with no errors in the test mode, answer 'N' to the test run question and you will be able to choose your own options in response to further questions.

TABLE 1

Highlight	HT Code	Keys to be pressed	Printer: Epson MX80-3	Epson RX80	Epson FX80	Juki 6100
Underline	128	Sh £4	x	x	x	x
Emphasised	129	Sh £5	x	x	x	
Superscript	130	Sh Ct £0	x	x	x	x
Subscript	131	Sh Ct £1	x	x	x	x
Enlarged	132	Sh Ct £2	x	x	x	x
Condensed	133	Sh Ct £3	x	x	x	x
Italic	134	Sh Ct £4		x	x	
Doublestrike	135	Sh Ct £5	x	x	x	x
Elite	136	Sh Ct £6		x	x	x
Pica	137	Sh Ct £7	x	x	x	
Lineandahalf	138	Sh Ct £8	x	x	x	x
Proportional	139	Sh Ct £9			x	
Pad		Sh Ct Cp	x	x	x	x
Backspace		Sh Ct <	x	x	x	x
Other features	Microspacing					x
	£ and #		x	x	x	x

Note: Sh = SHIFT Ct = CONTROL Cp = COPY fn = FUNCTION KEY n < = LEFT CURSOR

No error checking is performed this time since each user will choose his own options which alter the code and these cannot be predicted in advance.

The program in **Listing 1,** which is in BASIC 2, will create a machine code driver program. This new program will then be saved to disc or cassette for later use by the second program. You will be asked for your selected default options ie Epson interface type, Juki interface type, paper eject, screen mode, power up printer driver and the power up default status of justification, insertion and format.

The driver interface option must be selected for each of the drivers ie no printer (sets *FX5,0), parallel or serial.

The paper eject facility allows an extra sheet of paper to eject from the printer after your document has been printed. On some printers this saves the need to do a form feed in order to tear off the paper.

The default screen mode is chosen with regard to the display characteristics of your equipment. It is suggested that MODE 3 is used for monitors and MODE 6 or 7 for normal TV sets.

The printer driver selected at power up should, obviously, match the printer that you own. If your printer is not catered for, then DEFAULT should be chosen. Once operating in VIEW either of the other drivers may be entered by the normal PRINTER command, eg PRINTER JUKI will invoke the Juki driver, but it should be noted that PRINTER by itself is used to select the DEFAULT driver. The DEFAULT driver can be used with the facility described in 5 above to allow control of any printer. Other printer drivers can be loaded from cassette or disc provided they are not called EPSON or JUKI.

The option is given in this program to choose the default status (on or off) of justification, insertion and format so that you may tailor these facilities to your own requirements.

Listing 2 will be given next month. It is a program that has four functions. First, it does an intelligent search for the VIEW ROM ie it looks in socket 15 for VIEW and if it is not there it then looks in descending order until it finds it. Once found, an image is created starting at &3000.

Secondly, the program changes the contents of some locations which enables links to be made to the driver code produced by program one. It also changes the version number for A1.4 to B1.4. This latter change is done purely to identify which version of VIEW is in use.

Thirdly, the previously saved driver code is loaded back into memory at &6000 to combine with the existing code and finally the whole of the code from &3000 to &7000 is saved under the title VIEWMOD.

All that is then necessary is to program this file into an EPROM, or for those with 'sideways' RAM cards, to load the VIEWMOD file. **Listing 2** should be double checked since no error checking is built into this program and errors will be difficult to locate in VIEW.

LISTING 1(a)

```
   10 REM*******************************
   20 REM*                             *
   30 REM*      VIEW MODIFICATION      *
   40 REM*                             *
   50 REM*          PROGRAM 1          *
   60 REM*                             *
   70 REM*        in BASIC 2           *
   80 REM*                             *
   90 REM*            by               *
  100 REM*                             *
  110 REM*       P W G Simpson         *
  120 REM*                             *
  130 REM*           and               *
  140 REM*                             *
  150 REM*       B V Alderwick         *
  160 REM*                             *
  170 REM*******************************

  180 MODE7
  190 neffects=13
  200 base=&100
  210 repeatflag=base
  220 npass=base+1
  230 total=base+2
  240 tempx=base+3
  250 tempy=base+4
  260 pointer=base+5
  270 pb=base+6
  280 drivereffects=&40F
  290 flag=&420
  300 osnewl=&FFE7
  310 oswrch=&FFEE
  320 osasci=&FFE3
  330 osbyte=&FFF4
  340 osfind=&FFCE
  350 osword=&FFF1
  360 INPUT"IS THIS A TEST RUN (Y/N)",A$
  370 IF A$<>"Y" AND A$<>"N" THEN 360
  380 IF A$="N" THEN CLS:GOTO430
  390 epsonint=1:jukiint=1:pthrow=TRUE
  400 mode$="3":defprinter$="EPSON"
  410 just=0:insert=0:form=0
  420 testrun=TRUE:GOTO850
  430 testrun=FALSE
  440 PRINT"Enter EPSON interface type"'
 "0 = No printer"'"1 = Parallel interfac
e"'"2 = Serial interface"'
  450 A%=GET-48
  460 IF A%>2 OR A%<0 THEN PRINT;A%;" is
 an incorrect interface type":GOTO440
  470 epsonint=A%
  480 PRINT'"EPSON interface is type ";e
psonint
  490 IF NOT FNok THEN440
  500 CLS
  510 PRINT"Enter JUKI interface type"''
"0 = No printer"'"1 = Parallel interface
"'"2 = Serial interface"'
  520 A%=GET-48
  530 IF A%>2 OR A%<0 THEN PRINT;A%;" is
 an incorrect interface type":GOTO510
  540 jukiint=A%
  550 PRINT'"JUKI interface is type ";ju
kiint
  560 IF NOT FNok THEN 510
  570 CLS
  580 PRINT"Do you want an extra  blank
page  to be thrown when  printing has  f
inished  on the EPSON (Y/N) ?";
  590 A$=GET$
  600 IF A$<>"Y" AND A$<>"N" THEN PRINT:
GOTO580
  610 IF A$="Y" THEN pthrow=TRUE ELSE pt
hrow=FALSE
  620 CLS
  630 PRINT"In which  screen mode  do yo
u want VIEW to power up?"
  640 A$=GET$
  650 IF A$<"0" OR A$>"7" THEN PRINT'"In
correct mode"':GOTO630
  660 mode$=A$
  670 PRINT'"Default mode will be mode "
;mode$
  680 IF NOT FNok THEN 630
  690 CLS
  700 INPUT"Which printer driver  do you
 want VIEW to load on power up",A$
  710 IF A$="" THEN A$="DEFAULT"
  720 IF A$<>"EPSON" AND A$<>"JUKI" AND
A$<>"DEFAULT" THEN PRINT'"That printer d
river does not exist"':GOTO700
  730 defprinter$=A$
  740 IF A$="DEFAULT" THEN defprinter$="
"
  750 PRINT'"Default driver will be ";A$
  760 IF NOT FNok THEN 700
  770 CLS:INPUT"Do  you  want  JUSTIFICA
TION on or off? Enter 0 or 1 - "just
  780 just=just*255
  790 IF (just<>0) AND (just<>255) THEN
770
  800 CLS:INPUT"Do  you want  INSERTION
mode on or off? Enter 0 OR 1 - "insert
  810 insert=255*(1-insert)
  820 IF (insert<>0) AND (insert<>255) T
HEN 800
  830 CLS:INPUT"Do  you want  FORMAT  mo
de  on  or off? Enter 0 OR 1 - "form
  840 IF (form<>0) AND (form<>1) THEN 83
0
  850 CLS:PRINT"Assembling code"'
  860 DIM code &E00
  870 IF testrun THEN tr=1 ELSE tr=0
  880 FORI%=4 TO 6+tr STEP 2+tr
  890 PRINT"Pass ";(I%-4)/(2+tr)+1
  900 P%=&B000:O%=code
  910 [            OPT I%
  920              JMP prcode
  930              JMP stflag
  940              JMP endflag
  950              JMP stcomms
  960              JMP help
  970              JMP comsrch
  980              JMP hilites
  990              JMP prchar
 1000              JMP scen
 1010              JMP sccheck
 1020              JMP scfix
 1030              JMP forfix
 1040              JMP forfixa
 1050              JMP forfixb
 1060              JMP forfixc
 1070              JMP setmfix
 1080              JMP edcofix
 1090              JMP spltfix
 1100              JMP concfix
 1110              JMP confixa
 1120              JMP delcfix
 1130              JMP hdftfx3
 1140              JMP hdftfx2
 1150              JMP hdftfix
 1160              JMP gtmfix
 1170              JMP padfix
 1180              JMP hdftfx4
 1190              JMP formht
 1200              JMP reform
 1210              JMP tbform
 1220              JMP modes
 1230
 1240 .prcode  PHP:PHA:TXA:PHA:TYA:PHA
 1250          LDX#0:LDY#0
 1260 .prcode4 LDA prtable,X:BEQ prfound
 1270          LDA &7D8,Y
 1280          CMP#13:BEQ prfound
 1290          CMP#ASC"A":BMI prcode3
 1300          AND#&DF
 1310 .prcode3 CMP prtable,X:BNE prcode2
 1320          INX:INY
 1330          JMP prcode4
 1340
 1350 .prcode2 LDY#0:INX
 1360          LDA prtable,X:BNE prcode2
 1370          INX:INX:INX
 1380          LDA prtable,X:BNE prcode4
 1390          PLA:TAY:PLA:TAX:PLA:PLP
 1400          JSR &87B4
 1410          RTS
 1420
 1430 .prfound LDA &7D8,Y
 1440          CMP#13:BNE prcode2
 1450          LDA prtable,X:BNE prcode2
 1460          LDA 0:PHA:LDA 1:PHA
 1470          LDA prtable+1,X:STA 0
 1480          LDA prtable+2,X:STA 1
 1490          LDY#&11
 1500 .loop    LDA(0),Y:STA &400,Y
 1510          DEY:BPL loop
 1520          PLA:STA 1:PLA:STA 0
 1530          PLA:TAY:PLA:TAX:PLA:PLP
 1540          RTS
 1550
 1560 .prtable EQUS"EPSON":EQUB 0
 1570          EQUW epstab
 1580          EQUS"JUKI":EQUB 0
 1590          EQUW jukitab:EQUB 0
 1600
 1610 .epstab  JMP epsout
 1620          JMP epson
 1630          JMP epsoff
 1640          JMP epsshmi
 1650          JMP epsopt
 1660          JMP epseff
 1670
 1680 .jukitab JMP jukiout
 1690          JMP jukion
 1700          JMP jukioff
 1710          JMP jukshmi
 1720          JMP jukiopt
 1730          JMP jukieff
 1740
 1750 .errstop JSR osnewl
 1760          LDA#&FF:STA repeatflag
 1770          LDA#6:JSR &908A
 1780          LDA#0:TAY:JSR osfind
 1790          RTS
 1800
 1810 .getnum  LDA#0:STA total:STA npass
 1820          DEY
 1830 .getnum3 INY
 1840          LDA(&5),Y
 1850          CMP#32:BEQ getnum3
 1860          CMP#ASC",":BEQ getnum3
 1870 .getnum2 CMP#13:BEQ gnend
 1880          CMP#ASC"0":BCC gnend
 1890          CMP#ASC":":BCS gnend
 1900          SEC:SBC#48:PHA
 1910          LDA total:ASL A:PHA
 1920          ASL A:ASL A:STA total
 1930          PLA:CLC:ADC total
 1940          STA total
 1950          PLA:CLC:ADC total
 1960          STA total
 1970          INC npass:INY:LDA(&5),Y
 1980          JMP getnum2
 1990
 2000 .gnend   STY pointer
 2010          LDA npass:CLC:BNE gnend2
 2020          SEC
 2030 .gnend2  LDA total
 2040          RTS
 2050
 2060 .table EQUS"UNDERLINE":EQUW 0
 2070        EQUS"EMPHASISED":EQUW&100
 2080        EQUS"SUPERSCRIPT":EQUW&200
 2090        EQUS"SUBSCRIPT":EQUW&300
 2100        EQUS"ENLARGED":EQUW&400
 2110        EQUS"CONDENSED":EQUW&500
 2120        EQUS"ITALIC":EQUW&600
 2130        EQUS"DOUBLESTRIKE":EQUW&700
 2140        EQUS"ELITE":EQUW&800
 2150        EQUS"PICA":EQUW&900
 2160        EQUS"LINEANDAHALF":EQUW&A00
 2170        EQUS"PROPORTIONAL":EQUW&B00
 2180        EQUS"SENDPRINTER":EQUW&FF00
 2190        EQUB 0
 2200
 2210 .pronly  PHA:LDA#1:JSR oswrch
 2220          PLA:JSR oswrch
 2230          RTS
 2240
 2250 .esc     PHA:LDA#27:JSR pronly
 2260          PLA:JSR pronly
 2270          RTS
 2280
 2290 .pron    LDA#2:JSR oswrch
 2300          RTS
 2310
 2320 .proff   LDA#3:JSR oswrch
 2330          RTS
 2340
 2350 .effects SEC:SBC#128
 2360          CMP#neffects:BCC ef3
 2370          JSR errstop
 2380          BRK:EQUB 0
```

LISTING 1(b)

```
 2390 EQUS"Highlight code not supported"
 2400          EQUB 0
 2410
 2420 .ef3     TAX:CPX#2:BCC ef2
 2430          CPX#4:BCS ef2
 2440          PHA:LDX#neffects
 2450          JSR drivereffects
 2460          PLA:TAX
 2470          JSR drivereffects
 2480          LDX#neffects
 2490          LDA flag+10:BEQ ef2
 2500          JSR drivereffects
 2510          LDA#0:STA flag+10
 2520          LDX#10
 2530 .ef2     JSR drivereffects
 2540          RTS
 2550
 2560 .epsopt  LDY#0
 2570          RTS
 2580
 2590 .epson   LDA repeatflag:BEQ ep2
 2600          LDA#15:JSR oswrch
 2610          LDA#5:LDX#epsonint
 2620          LDY#0:JSR osbyte
 2630          JSR pron
 2640          LDA#ASC"@":JSR esc
 2650          LDA#ASC"R":JSR esc
 2660          LDA#0:JSR pronly
 2670          LDA#ASC"9":JSR esc
 2680          LDA#0:JSR pronly
 2690          LDA#0:LDX#16
 2700 .ep4     STA flag,X
 2710          DEX:BPL ep4
 2720          STA repeatflag
 2730          RTS
 2740
 2750 .ep2     JSR pron
 2760          LDA#ASC"8":JSR esc
 2770          RTS
 2780
 2790 .epsoff  LDA repeatflag:BEQ ep3
 2800 ]
 2810 IF pthrow THEN [OPT I%:LDA#12:JSR
pronly:]
 2820 [        OPT I%
 2830          LDA#ASC"@":JSR esc
 2840 .ep3     JSR proff
 2850          RTS
 2860
 2870 .epsshmi RTS
 2880
 2890 .epsout  PHP:STX tempx
 2900          STY tempy:PHA
 2910          CMP#96:BNE ep5
 2920          LDA#ASC"R":JSR esc
 2930          LDA#3:JSR pronly
 2940          LDA#35:JSR pronly
 2950          LDA#ASC"R":JSR esc
 2960          LDA#0:JSR pronly
 2970          JSR proff
 2980          LDA#96:JSR oswrch
 2990          JSR pron
 3000          JMP ep6
 3010
 3020 .ep5     CMP#128:BCC ep9
 3030          JSR effects
 3040          JMP ep6
 3050
 3060.ep9      JSR osasci
 3070.ep6      PLA:LDX tempx
 3080          LDY tempy:PLP
 3090          RTS
 3100
 3110.epseff   LDA flag,X:BNE ep8
 3120          LDA epon,X:JSR esc
 3130          STA flag,X
 3140          LDA epon2,X:BMI ep7
 3150          JSR pronly
 3160.ep7      RTS
 3170
 3180.ep8      LDA epoff,X:JSR esc
 3190          LDA epoff2,X:BMI ep20
 3200          JSR pronly
 3210.ep20     LDA#0:STA flag,X
 3220          RTS
 3230
 3240.epon     EQUS"-ESSW":EQUB 15
 3250          EQUS"4GMPAp#A"
 3260
 3270.epon2    EQUD &0100FF01
 3280          EQUD &FFFFFF01
 3290          EQUD &0112FFFF
 3300          EQUW &0508
 3310
 3320.epoff    EQUS"-FTTW75HPP2p#A"
 3330
 3340.epoff2   EQUD &FFFFFF00
 3350          EQUD &FFFF1200
 3360          EQUD &00FFFFFF
 3370          EQUW &0C08
 3380
 3390.jukiopt LDY#1
 3400          RTS
 3410
 3420.jukion   LDA repeatflag:BEQ ju2
 3430          LDA#15:JSR oswrch
 3440          LDA#5:LDX#jukiint
 3450          LDY#0:JSR osbyte
 3460          JSR pron
 3470          LDA#26:JSR esc
 3480          LDA#ASC"I":JSR pronly
 3490          LDA#0:LDX#16
 3500.ju3      STA flag,X
 3510          DEX:BPL ju3
 3520          LDA#0:STA repeatflag
 3530          RTS
 3540
 3550.ju2      JSR pron
 3560          RTS
 3570
 3580.jukioff JSR proff
 3590          RTS
 3600
 3610.jukshmi LDA#31:JSR esc
 3620          INX:TXA:JSR pronly
 3630          DEX
 3640          RTS
 3650
 3660.jukiout PHP:STX tempx
 3670          STY tempy:PHA
 3680          CMP#96:BNE ju4
 3690          LDA#ASC"I":JSR esc
 3700          JSR proff
 3710          LDA#96:JSR oswrch
 3720          JSR pron
 3730          JMP ju6
 3740
 3750 .ju4     CMP#128:BCC ju9
 3760          JSR effects
 3770          JMP ju6
 3780
 3790 .ju9     JSR osasci
 3800          CMP#&D:BNE ju6
 3810          LDA flag+1:BEQ ju5
 3820          LDA#0:STA flag+1
 3830          LDX#1:JSR jukieff
 3840 .ju5     LDA flag+7:BEQ ju6
 3850          LDA#0:STA flag+7
 3860          LDX#7:JSR jukieff
 3870 .ju6     PLA:LDX tempx
 3880          LDY tempy:PLP
 3890          RTS
 3900
 3910 .jukieff LDA flag,X:BNE ju8
 3920          LDA juon,X:JSR esc
 3930          STA flag,X
 3940          LDA juon2,X:BMI ju7
 3950          JSR pronly
 3960 .ju7     RTS
 3970
 3980 .ju8     LDA juoff,X:JSR esc
 3990          LDA juoff5,X:BMI ju20
 4000          JSR pronly
 4010 .ju20    LDA#0:STA flag,X
 4020          RTS
 4030
 4040 .juon    EQUS"EWDU":EQUW &1F1F
 4050          EQUS"1O":EQUB &1F
 4060          EQUW &1E1F:EQUS"11"
 4070          EQUB 30
 4080
 4090 .juon2   EQUD &FFFFFFFF
 4100          EQUD &FFFF0910
 4110          EQUD &FF0D0D0B
 4120          EQUW &0608
 4130
 4140 .juoff   EQUS"R&UDSS1&SS"
 4150          EQUB &1E:EQUS"11"
 4160          EQUB 30
 4170
 4180 .juoff5  EQUD &FFFFFFFF
 4190          EQUD &FFFFFFFF
 4200          EQUD &FF09FFFF
 4210          EQUW &0908
 4220
 4230 .stflag  PHP:PHA
 4240          LDA#&FF:STA repeatflag
 4250          PLA:PLP:JSR &8422
 4260          RTS
 4270
 4280 .endflag JSR &8BE1:PHP:PHA
 4290          LDA#&FF:STA repeatflag
 4300          PLA:PLP
 4310          RTS
 4320
 4330 .stcomms PHP:PHA:TXA:PHA:TYA:PHA
 4340          JSR escape
 4350          LDA#3:LDX#2:JSR osbyte
 4360          LDA#&FF:PHA
 4370 .byte3   PLA:CLC:ADC#1:PHA:TAX
 4380          LDY prname,X:BEQ byte2
 4390          LDA#138:LDX#0:JSR osbyte
 4400          JMP byte3
 4410
 4420 .byte2   JSR ichars
 4430          PLA
 4440          PLA:TAY:PLA:TAX:PLA:PLP
 4450          JSR osbyte
 4460          RTS
 4470
 4480 .prname  EQUS"OLD":EQUB 13
 4490          EQUS"*FX3":EQUB 13
 4500          EQUS"MODE "
 4510          EQUS mode$:EQUB 13
 4520          EQUS"PRINTER "
 4530          EQUS defprinter$:EQUB 13
 4540          EQUB 0
 4550
 4560 .comsrch LDA#ASC"C"
 4570          CMP &87:BNE cl2
 4580          LDA#ASC"L"
 4590          CMP &88:BEQ cbuilt
 4600 .cl2     JMP commend
 4610
 4620 .cbuilt  LDX#0:LDY#2
 4630 .cbuilt2 INY
 4640          LDA(&5),Y
 4650          CMP#65:BCC cbuilt2
 4660          CMP#123:BCS cbuilt2
 4670          STY pointer:DEY
 4680 .pr4     INY
 4690 .pr5     LDA table,X:BEQ found
 4700          LDA(&5),Y
 4710          CMP#13:BEQ found
 4720          CMP#ASC" ":BEQ found
 4730          BCC pr4
 4740          CMP#123:BCS pr4
 4750          CMP#ASC",":BEQ found
 4760          CMP#ASC"A":BMI pr3
 4770          AND#&DF
 4780 .pr3     CMP table,X:BNE pr2
 4790          INX:INY
 4800          JMP pr5
 4810
 4820 .pr2     LDY pointer:INX
 4830          LDA table,X:BNE pr2
 4840          INX:INX
 4850          LDA table,X:BNE pr5
 4860          JSR errstop
 4870          LDY pointer
 4880 .pr30    LDA(&5),Y
 4890          CMP#ASC" ":BEQ pr6
 4900          CMP#ASC",":BEQ pr6
 4910          CMP#13:BEQ pr6
 4920          JSR oswrch:INY
 4930          JMP pr30
 4940
 4950 .pr6     BRK
 4960          EQUB 0
 4970 EQUS" is not a valid CL command."
 4980          EQUB 0
 4990
 5000 .found   LDA(&5),Y
 5010          CMP#33:BCC fo2
```

LISTING 1(c)

```
5020          CMP#ASC",":BEQ fo2
5030          CMP#123:BCS fo2
5040          JMP pr2
5050
5060 .fo2     LDA table,X:BNE pr2
5070          INX
5080          LDA table,X:STY pointer
5090          BPL f2
5100 .f3      JSR getnum:BCS f4
5110          TAX:LDA &6D:BEQ f3
5120          TXA:JSR pronly
5130          JMP f3
5140
5150 .f2      ORA#&80:TAX
5160          LDA &6D:BEQ f4
5170          TXA:JSR chout
5180 .f4      LDY pointer:LDA(&5),Y
5190          CMP#13:BEQ commend
5200          DEY
5210 .f5      INY:LDA(&5),Y
5220          CMP#13:BEQ commend
5230          CMP#33:BCC f5
5240          CMP#ASC",":BEQ f5
5250          CMP#123:BCS f5
5260          STY pointer:LDX#0
5270          JMP pr5
5280
5290 .commend LDX#0:LDY#0
5300          RTS
5310
5320 .chout   JMP(&17)
5330
5340 .hilites BCC hili3
5350          CMP#&90:BCC hili2
5360          CMP#&9D:BEQ hili2
5370          CMP#&9E:BNE hili3
5380          PHA:LDA &37:BEQ bsperr
5390          DEC &37
5400 .bsperr  PLA
5410          RTS
5420
5430 .hili2   INC &37
5440 .hili3   RTS
5450
5460 .prchar  CMP#&90:BCS pc2
5470 .pc20    LDX#1:CLC
5480          RTS
5490
5500 .pc2     STY &84:LDY &6D:BEQ pc3
5510          CMP#&9D:BCC pc23
5520          BNE pc22
5530          LDA#32
5540          JMP pc24
5550
5560 .pc22    LDA#&8C
5570          JMP pc24
5580
5590 .pc23    SEC:SBC#&10
5600 .pc24    LDY &84
5610          JMP pc20
5620
5630 .pc3     PHA:LDA#135:JSR osbyte
5640          PLA
5650          CPY#7:BNE pc4
5660          CMP#&90:BNE pc5
5670          LDA#&5F
5680          JMP pc6
5690
5700 .pc5     CMP#&91:BNE pc7
5710          LDA#&2A
5720          JMP pc6
5730
5740 .pc7     CMP#&9D:BNE pc8
5750          LDA#&50
5760          JMP pc6
5770
5780 .pc8     CMP#&9E:BNE pc9
5790          LDA#&5B
5800          JMP pc6
5810
5820 .pc9     SEC:SBC#&62
5830 .pc6     LDY &84
5840          JMP pc20
5850
5860 .pc4     SEC:SBC#&10:LDY &84
5870          JMP pc20
5880
5890 .scen    JSR osbyte
5900 .        LDA#&E4:LDX#&BC:LDY#0
5910          JSR osbyte
5920 .ichars  LDX#119:STX pb
5930          LDA#&C:STA pb+1
5940          LDA#&FF:STA pb+2:STA pb+3
5950 .chloop  LDX pb:LDA invchars,X
5960          STA pb+4:LDA#6
5970          LDX#pb MOD 256
5980          LDY#pb DIV 256:JSR osword
5990          DEC pb:BPL chloop
6000          LDA#&F:JSR oswrch
6010          RTS
6020
6030 .sccheck CMP#&2D:BCS effkey
6040 .scch3   ORA#0:BNE scch2
6050          JSR escape
6060 .scch2   JMP &A12F
6070
6080 .escape  PHA:TXA:PHA
6090          LDA#&E1:JSR reskeys
6100          LDA#&E2:JSR reskeys
6110          LDA#&E3:JSR reskeys
6120          LDA#&E4:JSR reskeys
6130          PLA:TAX:PLA:LDY#0
6140          RTS
6150
6160 .reskeys LDX#0:LDY#0:JSR osbyte
6170          RTS
6180
6190 .effkey  CMP#&34:BCS effk2
6200          CMP#&2D:BNE effk4
6210          LDA#&92:JSR &97B1
6220          LDA#ASC"o":JSR &97B1
6230          LDA#&92:JSR &97B1
6240          LDA#ASC"C":JSR &97B1
6250          RTS
6260
6270 .effk4   PLA:PLA
6280          JMP &95AD
6290
6300 .effk2   CLC:ADC#&5E
6310          CMP #&9D:BNE effk3
6320          STA &36
6330          PLA:PLA:LDA &36
6340          JMP &95E8
6350
6360 .effk3   JMP &97B1
6370
6380 .scfix   STY &37
6390          LDA(&85),Y:BPL scfix3
6400          CMP#&90:BCC scfix2
6410          AND#0
6420          RTS
6430
6440 .scfix2  LDA(&85),Y
6450 .scfix3  RTS
6460
6470 .forfix  CMP#&90:BCS ff2
6480          ORA#0:BPL ff2
6490          JMP &A2CE
6500
6510 .ff2     JMP &A2DB
6520
6530 .forfixa LDA(&9),Y
6540          CMP#&90:BCS ff5
6550          ORA#0:BPL ff5
6560          JMP &94AD
6570
6580 .ff5     JMP &94B0
6590
6600 .forfixb CMP#&90:BCS ffixb3
6610          CMP#&7F:BCS ffixb2
6620 .ffixb3  JMP &952A
6630
6640 .ffixb2  JMP &9563
6650
6660 .forfixc LDA(&5),Y
6670          CMP#&90:BCS ffixc2
6680          LDA(&5),Y
6690 .ffixc2  PHP
6700          JMP &A318
6710
6720 .setmfix LDA(&5),Y
6730          CMP#&90:BCS smfix2
6740          LDA(&5),Y
6750 .smfix2  PHP
6760          JMP &A54B
6770
6780 .edcofix LDA(&85),Y
6790          CMP#&90:BCS ecf2
6800          LDA(&85),Y:BPL ecf2
6810          JMP &8C43
6820
6830 .ecf2    JMP &8CB6
6840
6850 .spltfix LDA(&5),Y
6860          CMP#&90:BCS splfix3
6870          LDA(&5),Y:BMI splfix2
6880 .splfix3 RTS
6890
6900 .splfix2 PLA:PLA
6910          RTS
6920
6930 .concfix LDA(&9),Y
6940          CMP#&90:BCS confix3
6950          LDA(&9),Y:BMI confix2
6960 .confix3 RTS
6970
6980 .confix2 PLA:PLA
6990          RTS
7000
7010 .confixa CMP#&90:BCS cfixa2
7020          TAX:BPL cfixa2
7030          PLA:PLA
7040          RTS
7050
7060 .cfixa2  DEY:RTS
7070
7080 .delcfix LDA(&5),Y:BPL delcfx2
7090          CMP#&90:BCS delcfx2
7100          RTS
7110
7120 .delcfx2 PLA:PLA:RTS
7130
7140 .hdftfx3 INY
7150          LDA#0:STA total
7160          LDA(&89),Y:BPL hdftfx3
7170          CMP#&9F:BCC hdftfx3
7180          RTS
7190
7200 .hdftfx2 LDA(&87),Y
7210          CMP#&1C:BEQ hff22
7220          CMP#&1D:BEQ hff22
7230          CMP#&9D:BEQ hdft3
7240          CMP#&9E:BNE hff23
7250          JSR hff22
7260 .hff23   LDA(&87),Y:BPL hdft3
7270          CMP#&9F:BCS hdft2
7280 .hff22   INC total
7290          RTS
7300
7310 .hdft2   STX &84:TAX:LDA &84
7320          SEC:SBC total:STA &84
7330          PLA:PLA:TXA:LDX &84
7340 .hdft3   RTS
7350
7360 .hdftfx4 LDY#0:LDA &84
7370          CLC:ADC total:TAX
7380          RTS
7390
7400 .hdftfix STX tempx:JSR &9EBA
7410          LDX tempx:JSR &9077
7420          CMP#&80:BCC hdftfxa
7430          LDA &670,Y
7440          CMP#&9D:BEQ hdftfxa
7450          CMP#&9E:BNE hdftfxb
7460          JSR hdftfxb
7470 .hdftfxb DEC &37
7480 .hdftfxa RTS
7490
7500 .gtmfix  LDA(&9),Y:BPL gtmf5
7510          CMP#&90:BCS gtmf5
7520          RTS
7530
7540 .gtmf5   PLA:PLA:LDA(&9),Y
7550          JMP &A456
7560
7570 .formht  STA(&3),Y:INY
7580          CMP#&90:BCC fht2
7590          CMP#&9D:BEQ fht2
7600          BCC fht3
7610          INC &80
7620          JMP fht3
7630
7640 .fht2    CMP#&1C:BEQ fht3
7650          CMP#&1D:BEQ fht3
```

LISTING 1(d)

```
7660          RTS
7670
7680 .fht3    INC &80:RTS
7690
7700 .reform  STY tempy:LDY#0:STY total
7710          LDA(3),Y:CMP#9:BNE ref22
7720          LDA(7),Y
7730          CMP#ASC"*":BNE ref7
7740          INY:INC total
7750 .ref7    LDA(7),Y:INY
7760          CMP#ASC"*":BEQ ref6
7770          INC total:CMP#13:BNE ref7
7780          LDA#2:STA total
7790 .ref6    DEC total:LDY#1
7800 .ref22   LDA(3),Y:CMP#&9D:BEQ ref2
7810          BCS ref4
7820          CMP#&90:BCS ref3
7830          CMP#9:BEQ ref2
7840          CMP#&B:BEQ ref8
7850          CMP #ASC" ":BCC ref3
7860 .ref2    INC total
7870 .ref3    INY:CPY tempy:BCC ref22
7880          LDY total:BEQ ref5
7890          DEY
7900 .ref5    CPY &3C
7910          RTS
7920
7930 .ref4    LDA total:BEQ ref3
7940          DEC total
7950          JMP ref3
7960
7970 .ref8    TYA:PHA
7980          LDA total:PHA
7990 .ref20   LDA(7),Y:INY
8000          CMP#ASC">":BEQ ref9
8010          INC total
8020          CMP#13:BNE ref20
8030          PLA:PHA
8040          INC total
8050 .ref9    PLA:PLA:TAY:DEC total
8060          JMP ref3
8070
8080 .tbform  JSR &9EA8:DEX
8090          RTS
8100
8110 .modes   LDA#form:STA &4F
8120          LDA#just:STA &50
8130          LDA#insert:STA &51
8140          RTS
8150
8160 .padfix  BPL pf2
8170          SEC:SBC#&10
8180 .pf2     JSR oswrch
8190          RTS
8200
8210 .invchars EQUD&FFFFFFFF
8220 EQUD&FF8181FF:EQUD&C381E7FF
8230 EQUD&FFFFE781:EQUD&81C3E7FF
8240 EQUD&FFE7E7E7:EQUD&E7E7E7FF
8250 EQUD&FFE7C381:EQUD&F99981FF
8260 EQUD&FF819FB1:EQUD&83F981FF
8270 EQUD&FF81F9F9:EQUD&9F9F9FFF
8280 EQUD&FFF38193:EQUD&819F81FF
8290 EQUD&FF8199F9:EQUD&819F81FF
8300 EQUD&FF819999:EQUD&F99981FF
8310 EQUD&FFF9F9F9:EQUD&C399C3FF
8320 EQUD&FFC39999:EQUD&999981FF
8330 EQUD&FFF9F981:EQUD&0:EQUD&0
8340 EQUD&999981FF:EQUD&FF9F9F81
8350 EQUD&8FE3F9FF:EQUD&FFF9E38F
8360
8370 .help    JSR &A0EE:PHP
8380          CMP#&1B:BEQ lo2
8390          JMP end2
8400
8410 .lo2     PHA:TXA:PHA:TYA:PHA
8420          JSR ichars
8430          LDA 0:PHA:LDA 1:PHA
8440          LDA#helpmes MOD 256:STA 0
8450          LDA#helpmes DIV 256:STA 1
8460          LDY#0:LDA#12:JSR oswrch
8470          LDA#14:JSR oswrch
8480 .lo6     LDA(0),Y:BEQ end
8490          BMI lo7
8500          CMP#12:BEQ lo8
8510 .lo3     JSR osasci
8520          INY:BNE lo6
8530          INC 1
8540          JMP lo6
8550
8560 .lo7     STY tempy:JSR pc3
8570          LDY tempy
8580          JMP lo3
8590
8600 .lo8     STY tempy:LDA#135
8610          JSR osbyte:CPY#3:BEQ lo9
8620          CPY#6:BCS lo9
8630          LDY#7:LDA#13
8640 .lo4     JSR osasci:DEY:BNE lo4
8650 .lo9     LDY tempy:LDA#12
8660          JMP lo3
8670
8680 .end     TSX:LDA#0:STA &11B,X
8690          PLA:STA 1:PLA:STA 0
8700          PLA:TAY:PLA:TAX:PLA
8710 .end2    PLP
8720          RTS
8730
8740 .helpmes EQUS"      HIGHLIGHT COM
MANDS"
8750 EQUW&0D0D
8760 EQUS"  FUNCTION      KEYS     CHARA
CTER":EQUB 13
8770 EQUS"
":EQUB 13
8780 EQUS"Underline       sh+f4          "
8790 EQUW&D90
8800 EQUS"Emphasised      sh+f5          "
8810 EQUW&D91
8820 EQUS"Superscript     sh+ct+f0       "
8830 EQUW&D92
8840 EQUS"Subscript       sh+ct+f1       "
8850 EQUW&D93
8860 EQUS"Enlarged        sh+ct+f2       "
8870 EQUW&D94
8880 EQUS"Condensed       sh+ct+f3       "
8890 EQUW&D95
8900 EQUS"Italic          sh+ct+f4       "
8910 EQUW&D96
8920 EQUS"Doublestrike    sh+ct+f5       "
8930 EQUW&D97
8940 EQUS"Elite           sh+ct+f6       "
8950 EQUW&D98
8960 EQUS"Pica            sh+ct+f7       "
8970 EQUW&D99
8980 EQUS"Lineandahalf    sh+ct+f8       "
8990 EQUW&D9A
9000 EQUS"Proportional    sh+ct+f9       "
9010 EQUW&D9B
9020 EQUS"Pad            sh+ct+copy      "
9030 EQUW&D9D
9040 EQUS"Backspace      sh+ct+csrlft    "
9050 EQUW&D9E
9060 EQUS"Degrees C         ct+f9         "
9070 EQUB&92:EQUS"o":EQUB&92:EQUS"C"
9080 EQUB 13
9090 EQUS"Sendprinter   CL command only"
9100 EQUW&0D0D:EQUD&0D0D0D0D
9110 EQUB12
9120 EQUS"  POWER UP STATUS AND STORED
COMMANDS"
9130 EQUW&0D0D:]
9140 PROCst("Formatting","form=0")
9150 PROCst("Justification","just=0")
9160 PROCst("Insertion","insert<>0")
9170 PROCint("EPSON",epsonint)
9180 PROCst("EPSON Pagethrow","pthrow")
9190 PROCint("JUKI",jukiint)
9200 IFdefprinter$=""THEN A$="DEFAULT"
ELSE A$=defprinter$
9210 [OPT I%
9220 EQUS"Screen mode - "+mode$:EQUB13
9230 EQUS"Printer driver - "+A$
9240 EQUW&0D0D
9250 EQUS"  CL fuction,function..."
9260 EQUB 0:]
9270 NEXT
9280 PRINT'"Completed"''"Code located f
rom &";~code;" TO &";~O%
9290 IF NOT testrun THEN 9520
9300 PRINT'"Starting checksums"
9310 DATA 16184,16506,11476, 8148,12615
9320 DATA 12805,12766,12863,13261,13258
9330 DATA 15780,15519,15652,15080,13550
9340 DATA 13949,14193,17441,19851,12420
9350 DATA  9579, 8722, 8562, 9423, 9879
9360 DATA 11185
9370 E%=0
9380 FOR I%=code TO O%-128 STEP 128
9390 S%=0
9400 FOR J%=0 TO 127
9410 S%=S%+(I%?J%)
9420 NEXT
9430 READ T%
9440 IF S%=T% THEN 9480
9450 S%=I%-code+&B000
9460 PRINT"ERROR IN BLOCK FROM &";~S%;"
TO &";~S%+&7F
9470 E%=1
9480 NEXT
9490 IF E%=0 THEN PRINT"Checksums OK" E
LSE PRINT'"MAKE SURE CHECKSUM NUMBER IN
THE DATA"'"STATEMENT IS CORRECT BEFORE C
HECKING"'"THE SOURCE CODE!"
9500 IF testrun THEN PRINT'"Test run co
mpleted"
9510 IF E%<>0 OR testrun THEN END
9520 PRINT'"Saving code as file called
VMODS":OSCLI("*SAVE VMODS "+STR$~(code)+
" +E00")
9530 END
9540
9550 DEFFNok
9560 PRINT'"Is this correct (Y/N) ? ";
9570 A$=GET$
9580 IFA$<>"Y"ANDA$<>"N"THEN9560
9590 PRINT'
9600 IFA$="Y"THEN =TRUE ELSE =FALSE
9610
9620 DEF PROCst(X$,Y$)
9630 [          OPT I%
9640            EQUS X$+" - "
9650 ]
9660 IF EVAL(Y$) THEN [OPTI%:EQUS"on":E
QUB13:] ELSE [OPTI%:EQUS"off":EQUB13:]
9670 ENDPROC
9680
9690 DEF PROCint(X$,X)
9700 [          OPT I%:EQUS X$+" Interfac
e destination - ":]
9710 ON X+1 GOTO 9720,9730,9740
9720 [          OPT I%:EQUS"None":EQUB13:
]:ENDPROC
9730 [          OPT I%:EQUS"Parallel":EQU
B13:]:ENDPROC
9740 [          OPT I%:EQUS"Serial":EQUB
13:]:ENDPROC
>
```

DISK DRIVE SPECIAL

DO IT ON A DISK

Disk drives leave cassette recorders standing when it comes to speed and storage capacity. Liz Gregory delves into their inner workings, and examines the pros and cons of buying a drive.

As more and more users demand greater memory capacity and reliability, disk drives are increasing in their popularity as peripherals. There is now a greater variety of devices on the market and prices are falling as competition grows. Thus the luxury which used only to be afforded the business user has now become viable for the home micro owner.

Disk drives are mass storage devices which provide read/write non-volatile memory. Volatile memory includes RAM which disappears when the computer is switched off and ROM which cannot be written to. The only alternatives to disk drives are EPROMs which are erasable, programmable ROMs, CMOS-backed RAM where memory storage is supported by batteries when the computer is switched off, and cassette tape recorders.

Cassettes are adequate especially if users simply want to load games and shorter programs. If this is the case then the price of a disk drive cannot really be justified. However the computer does pick up interference fairly easily and data may just be distorted, as a micro is more distinguishing than the ear.

The advantages of disk drives over the common cassette recorder are evident. This is hardly surprising, as the former is a peripheral which is specifically designed for use with a computer whereas cassettes were never intended for this purpose. A drive both loads and stores information very quickly in comparison with cassettes which are slow and require loading from the start of the tape. With a drive, a particular piece of information is found virtually immediately as data can be accessed randomly on disk, whereas serial recording on tapes does not allow access at any particular point. Therefore a tape has to be wound on until the required program is reached and this involves a lot of wasted time.

The amount that can be stored on a single disk is considerably higher than that which can be recorded on a tape. Disks are much neater to store and can easily be sent through the post. If you have dual drives, disks can be copied with greater ease and should always be backed up, especially if the program is important. Even so, disks are not an ideal medium as too many deletions and insertions can cause bad sectors which prevent the acceptance of data.

Floppy disks are not as vulnerable as cassettes, but are still susceptible to ash, fingerprints etc. and must never be put near magnetic fields such as those emanating from telephones, loudspeakers, electric typewriters etc. Such contact will damage the recorded matter. They do have a protective casing but still must not be mishandled or subjected to extreme variations in temperature. And, although data stored is well protected, it is still better to re-record every couple of years or so.

Disks, like cassettes, are hardware specific. Disk drives operate under instruction from the host computer's operating system (OS), and for this reason most software on disk cannot be used on other machines unless they have an identical OS.

'. . . the luxury which used only to be afforded the business user has now become viable for the home micro owner'

Floppy disk drives

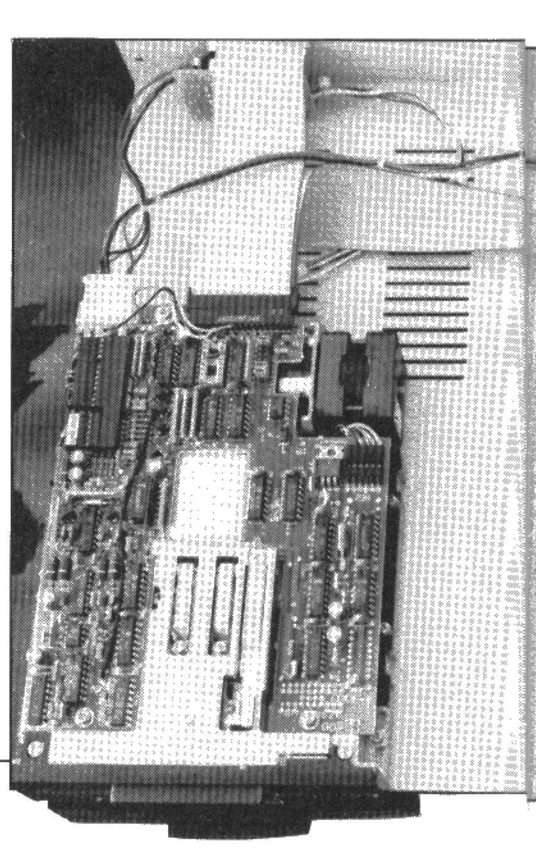
The majority of home micros use floppy disk drives. Invented by IBM in 1973 the US companies were quick to develop these and 8″ drives became readily available. When it became apparent that these were not fast or compact enough, 5¼″ drives came to the fore from companies like Shuggart. Nowadays, 8″ drives are normally associated with business machines and the 5¼″ drives are the most popular using their mini-floppies or diskettes.

The floppy disk itself is plastic, coated with a magnetic surface; that is ferric oxide; and is protected by a cardboard case. Some disks also have an additional protective "hub" around the centre (see **Figure 1**). All disks have a notch which can be "tabbed", that is protected by a metallic tab, so that it cannot be written upon. This operation can prove very useful when copying or transferring files from disk to disk.

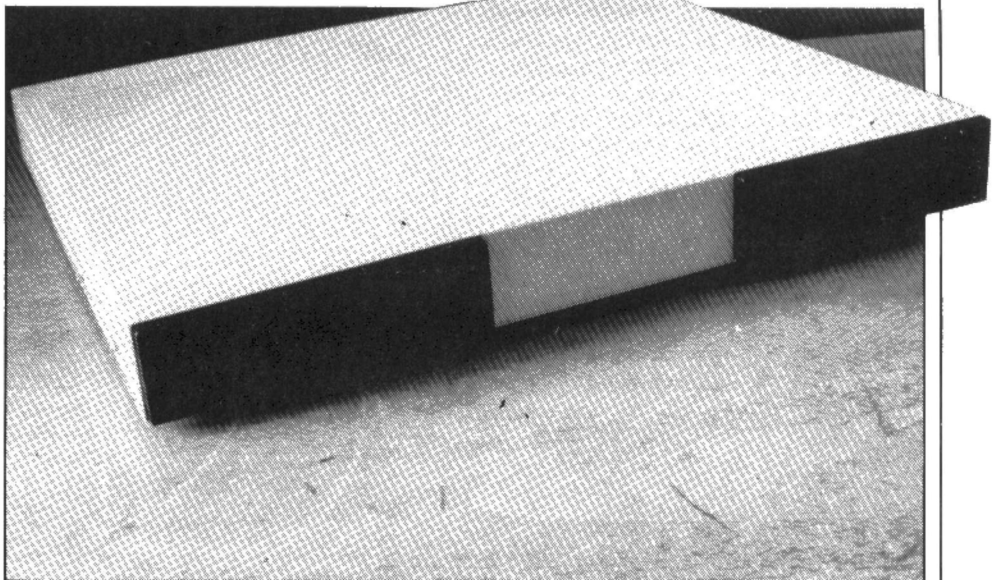
Data is distinguished by putting it in 'files' so that easy identification can be made. These 'files' consist of long strings of bytes and may be of differing types depending upon the information that they contain. In the case of a CP/M operating system for example, files may be 'document' or 'non-document' and each file is given a name. This should consist of two parts, as each name is given a qualifying extension to indicate to the operating system the exact nature of its material. Thus the system will know where to place it on the disk. These names are recorded in the file directory which is contained in a specific area of the disk, sometimes on the outside but more usually in the centre so that it can be rapidly and readily accessed.

Disk division

Before they can be used disks have to be prepared for use so that data might be recorded. This operation of 'formatting' consists of the disk being magnetically divided and is a verification process which with some machines can be watched on a



DRIVE



VDU; as the various tracks are being acknowledged.

These tracks are not like the grooves of a record, however, but are concentric circles all containing the same amount of data

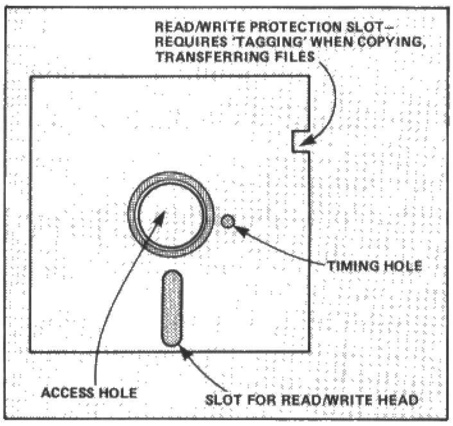
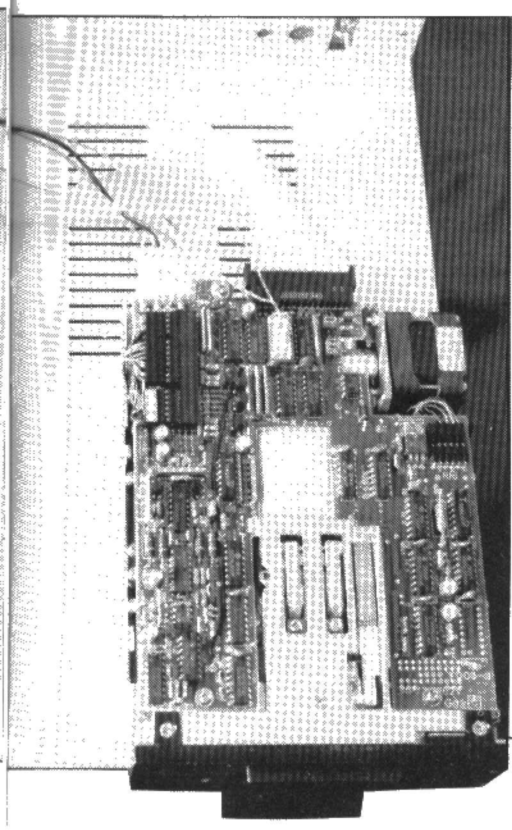


Figure 1. A soft sector disk complete with protective cardboard cover.

(see **Figure 2**). Disks might be 80, 45 or 40 track. Of these, 80 track disks require greater precision of the head because the distance between the tracks is smaller and data is more condensed.

These tracks are again subdivided into sectors. A typical track will have perhaps 10 sectors each of which will contain 256 bytes unless the disk is double density when 512 bytes per sector will be stored.

These sectors determine the nature of a disk. A hard sectored disk features a series of holes punched towards the centre to indicate the beginning of a new sector. Most disk drives for home micros are, however, soft-sectored and have only one hole which is used as a kind of reference for the computer and indicates the beginning of each track.



Again these sectors are divided into parts. Each has a 'header' which contains information about the track and sector number. A useful feature is the CRC, a

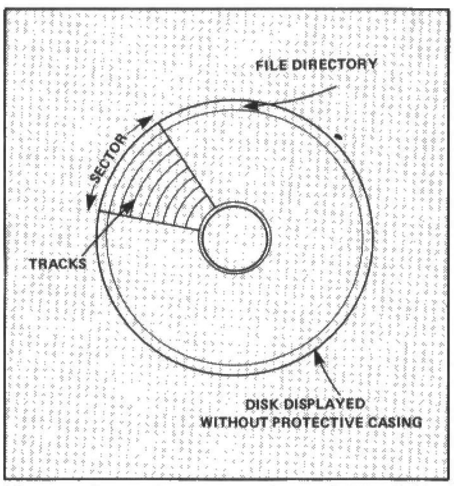


Figure 2. The different parts that a disk is divided into.

'cyclic redundancy check' which ensures that the file is readable and/or calcuable. Another phase is a sector gap which is necessary to prevent any overlapping or overwriting of data.

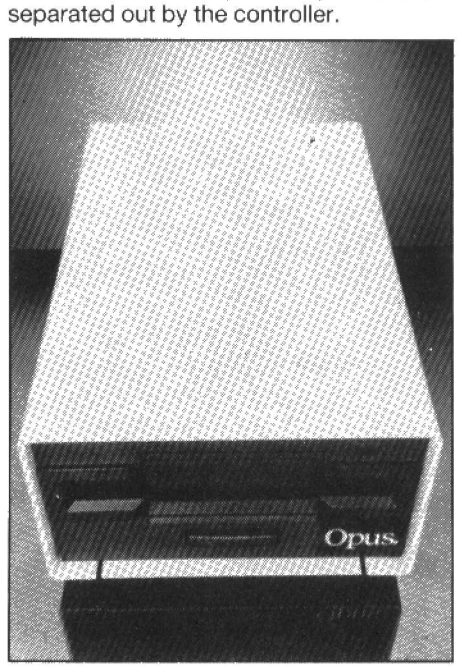
Operation

In order to see exactly how a disk drive functions both in a mechanical and electronic sense, we must look at both the hardware and software involved. The actual drive consists of a holder for the disk and a motor which drives the disk via a spindle at between 300-360 revolutions per minute. A stepper motor powers the mechanism which holds the electromagnetic head (see **Figure 3**). This is governed by an electronic controller situated at the computer's end.

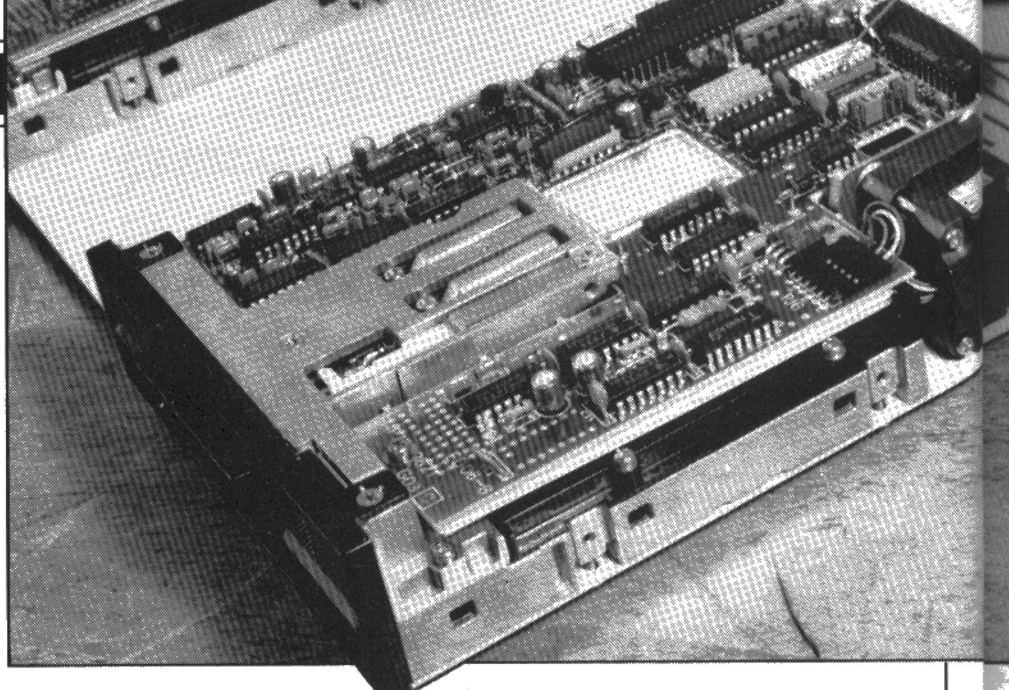
As this read/write head moves over the surface of the rotating disk it has to find the correct track. It picks up the electrical signals sent from the computer via the disk controller. This controller translates data and sends it, via an amplifier, to the head which produces the fields of magnetism which are recorded on the disk when data is being written. When information already stored is being read, then the process is reversed and data is translated back into a form that the computer will understand.

On a single-sided disk this head is kept in contact with the disk by a pressure pad on the opposing side of the medium. With a double-sided drive, a second head replaces this pad and therefore the capacity of a disk may be increased.

Data is synchronised because along with every byte of data a timing pulse is sent so that it is kept in step. These are separated out by the controller.



FEATURE



The computer requires a fixed signal from the head so that it has an idea when the beginning of each track occurs. This is where the small index hole of a soft-sectored disk comes in. A beam of light is shone down on the disk and as it rotates the hole lets lights through and a photosensor picks it up. This is therefore used as a reference point.

When purchasing a disk drive, one must consider the nature of the operating system of the computer. This enables the interfacing of machines to peripherals and is a program written in machine-code. Most systems for home micros come in ROM form and are Disk Operating Systems (DOS).. Normally all DOS have Disk Filing Systems which interpret disk commands that are typed in and converted into signals so that the disks might be operated. The only problem with interfaces like Acorn's 0.90 which includes a DFS is that a fair amount of RAM is used by them. They are also quite an expensive essential which has to be bought alongside a disk drive.

Name	Company	Type	Tracks	Sides	Capacity	Price
TRK1	Tech Op	Dual	40/80	Single	—	**£257.00**
TRK2	Tech Op	Dual	40/80	Double	—	**£349.00**
Slimline	Twillstar	Single	40	Single	100k	**£146.95**
Slimline	Twillstar	Dual	40/80	Single	200k	**£286.95**
Slimline	Twillstar	Dual	40/80	Double	400k	**£486.00**
Slimline	Twillstar	Single	80	Double	400k	**£268.91**
Slimline	Twillstar	Dual	80	Double	400k	**£503.00**
	AMS	Single	40	Double	200k	**£195.00**
	AMS	Double	40	Double	400K	**£347.00**
FD55A	Viglen	Single	40	Single	100k	**£134.79**
FD55E	Viglen	Single	40/80	Single	200k	**£169.57**
FD55B	Viglen	Dual	40	Double	400k	**£400.00**
FD55F	Viglen	Dual	40/80	Double	800k	**£415.66**
	Carson Developments	Single	40	Single	100k	**£140.00**
	Carson Developments	Double	40	Double	200k	**£170.00**
	Carson Developments	Double	80	Double	800k	**£400.00**
PSD1	Pace	Single	40	Single	100k	**£149.00**
PSD3	Pace	Single	40/80	Double	400k	**£256.00**
PDD1	Pace	Dual	40	Single	200k	**£294.00**
PDD3	Pace	Dual	40/80	Double	800k	**£417.00**
	Micro Resources	Single	45	Single	100k	**£130.00**
	Micro Resources	Dual	45	Single	100k	**£216.00**
CS100	Cumana	Single	40	Single	100k	**£169.00**
CD200	Cumana	Dual	40	Single	200k	**£305.00**
CD400/s	Cumana	Dual	40/80	Single	400k	**£469.00**
CD800/S	Cumana	Dual	40/80	Double	800k	**£499.00**
	Watford	Single	40	Double	100k	**£139.00**
	Watford	Single	40/80	Double	400k	**£215.00**
	Watford	Double	80	Double	800k	**£499.00**
	Technomatic	Single	40	Single	100k	**£150.00**
	Technomatic	Dual	40/80	Single	400k	**£400.00**
	Technomatic	Dual	80	Double	800k	**£420.00**
	Midwich	Single	40	Single	100k	**£153.00**
	Midwich	Dual	40/80	Single	400k	**£479.00**
	Opus Supplies	Single	40	Single	100k	**£149.00**
	Opus Supplies	Dual	40/80	Single	400k	**£514.00**
	Opus Supplies	Dual	80	Dual	800k	**£430.00**

Costs involved

Once you have decided to invest in a disk drive, you must carefully consider just how much it is going to be utilised and how important the data is that you wish to store. A single drive, which uses single density disks can be adequate for the user who does not require great volume or backing facilities. In which case, the peripheral works out as fairly expensive but prices are falling all the time, as companies like Carson Developments, Midwich and Opus Supplies all break the sub £150 barrier. If the drive has two heads then both sides of the disk can be used at only a small price increase; for example Microware's Mitsubishi model costs just over £200. An 80 track drive becomes expensive and it is perhaps better to opt for the wider option of 40/80 track devices like, for example, some of Cumana's disk drives which give greater scope although cost slightly more.

More serious users might find that a cheaper dual drive is better purchased at the outset as the cost of upgrading at a later stage is quite high. The advantages of having a dual drive are manifold as the difficulties of copying and backing-up disappear. Some dual drives like AMS' 3″ version offer double density and greater flexibility but are quite expensive at around £350. If the machine operates on CP/M then facilities like PIP (peripheral interface program) can be very useful. Here a file can be transferred from one disk to another which can be very handy especially when forms need to be "pipped" across. Notwithstanding this, a double density, dual disk drive offering 800K of memory can cost as much as £500 which is more than the price of most home micros.

The future

It is becoming apparent that people want

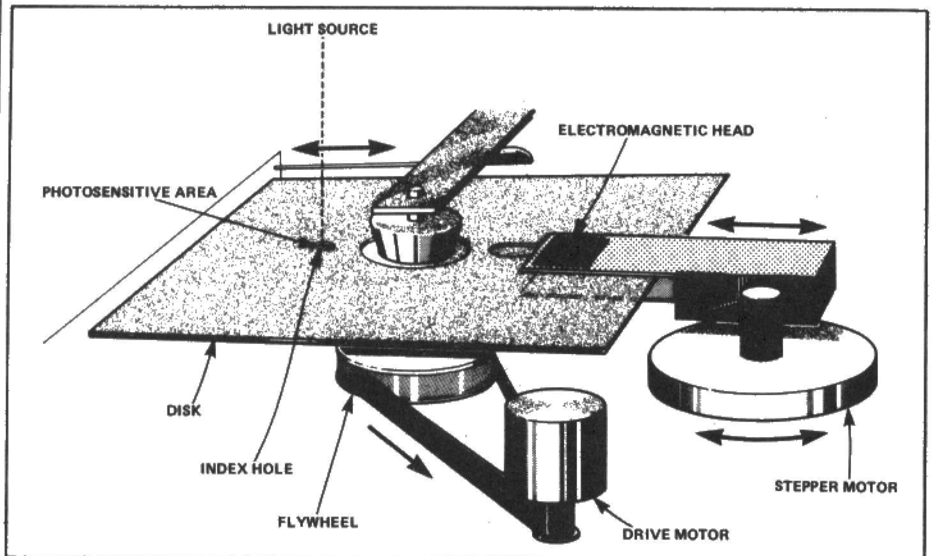


Figure 3. The mechanisms of the disk drive and head.

to use their machines for far more than just games. As drives fall in price we could see the use of cassettes die out completely. It will only remain for the user to decide exactly what kind of drive he needs and can afford. As for the question of size, the indications are that 3" drives will become more popular. However, in many cases, although 3 or 3½" drives are neater and more compact, the reliability and unimportance of size will probably keep 5¼" disk drives the first choice of home users.

As to the future of disk drives as peripherals, there has been quite a lot of talk recently about other forms of memory media. Bubble memory using tiny modules of magnetic substance, has already been incorporated in business machines with much success. This is extremely expensive and is a long way from the home user at the moment. Laser disks offer huge storage capacities and are exciting prospect, as they will not decay. Yet these are again hugely expensive. It will, therefore, be quite some time before disk drives cease to have a function for the home user and are thus a very viable prospect as a peripheral. ■

USEFUL ADDRESSES

AMS
Woodside Technology Centre
Green Lane
Appleton
Cheshire WA4 5NG

Carson Developments
84 Highfield Road
Romford
Essex

Cumana
Pines Trading Estate
Broad Street
Guildford
Surrey GU3 3BH

Midwich
Rickinghill House
Hinderclay Road
Rickinghill
Suffolk IP22 1HH

Opus Supplies
158 Camberwell Road
London SE5

Microware
Stanhope House
Fairbridge Road
London N19

DISK DRIVE SPECIAL

RANDOM ACCESS

Adam Denning has some ordered thoughts on the implementation of random access files on disk based BBC micro systems.

Owners of BBC Micros with disc drives will know that they can implement random access files, but may wonder what they are and how to use them.

A random access file is a file on disc that can be read from or written to at any desired place in the file. This sort of thing is very useful when you want to store information for later sorting or querying. Database systems are great users of random access files – or they should be!

To create a random access file, we need to give the file a name. We'll use RANDOM throughout the rest of this article. As this file is new, initially we have to open it just for writing to only (because of a quirk of the BBC Micro). We can do this in Basic like this:

```
S%=OPENOUT("RANDOM")
```

This does two things. It checks the disc to see if a file called RANDOM already exists. If it does, the old file is deleted (watch this!) and a new one of the same length is created, otherwise a completely new file is opened with a nominal length of &4000 bytes (16K).

Secondly, it produces a 'handle' for the open file and puts it into S%. The handle is the number the operating system uses to identify the file. The actual number is not relevant; however, we must not forget it. More particularly, if it is 0 then the file could not be opened for some reason. The disc might have been full or the write-protect tab might still be attached.

Having opened RANDOM for writing we are now in the position of being able to write to it, but as we are intending to treat it as a random access file, we'd better make it one now. To do this we have to go through the odd process of closing the file and then opening it again using OPENUP. So, our random access file creation program now becomes:

```
10  S%=OPENOUT ("RANDOM"):
    CLOSE#S%:
    S%=OPENUP("RANDOM")
```

Notice how the handle was used in the CLOSE statement so that the operating system knew which file it had to close.

Now we can use the Basic keywords BGET#, BPUT#, PTR# and EXT# to manipulate our file. But first some basic information. After opening our file for writing, it was given a length of &4000, but we closed it straightaway, without writing anything to it. The Beeb is pretty sensible here, so when we closed the file it changes its length to zero for us. When we opened it again with OPENUP, the file arguments were not changed – so it is still 0 bytes long. This can be found by typing PRINT EXT#S%. EXT# simply gives us the length of the file, and we cannot change it except by writing to the file.

If we now type PRINT PTR#S%, the answer will again by 0. PTR# is the place in the file where the next byte will be written to or read from, and we can ask the computer what its value is (as we just did) or we can tell it where to put it. PTR# starts from 0, so the 100th byte in the file is pointed to when PTR# equals 99.

"... a random access file is a file on disc that can be read from or written to at any place in the file ..."

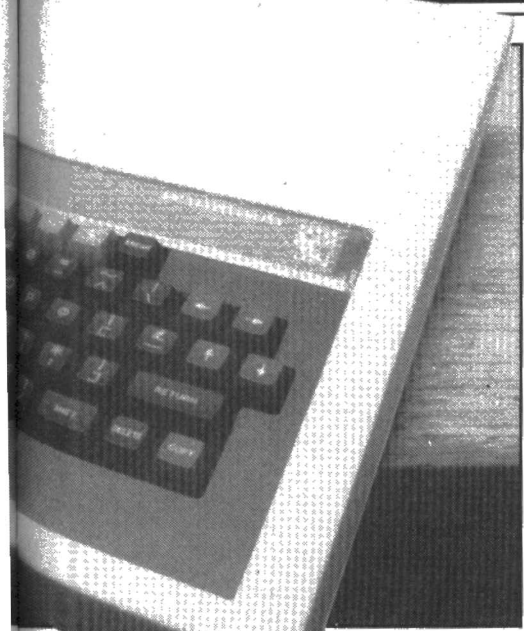
To read a byte from a file, we use BGET#, but as we haven't written anything to it all we'll get if we try it now is a report saying 'End of file'. So let's write something. We'll put our name at the 100th position onwards. Add lines 20 and 30 to the program above

```
20  INPUT"What is your name? "A$
30  PTR#S%=99:FORA%=1 TO
    LENA$:BPUT#S%,
    ASC(MID$(A$,A%,1)):NEXTA
```

To check that our name is really there, we can move the pointer back and use BGET# to read our name back:

```
40  PTR#S%=99:REPEAT A%=
    BGET#S%:IF A%<12 OR A%
    VDUA%
50  UNTIL A%>12 OR A%<32
```

This is all pretty basic stuff, so to really get things going we're going to do it all in machine code. Luckily the designers of the BBC Micro made things easy for us by giving details of operating system routines to do all this. The important ones here are OSFIND (address &FFCE) and OSARGS (address &FFDA). OSFIND is used by OPENOUT, OPENIN, OPENUP and CLOSE and works like this; by setting the X and Y registers to point to a filename followed by a carriage return (ASCII 13) stored in memory, and A contains one of three values. If A=&40 the file is opened for read only (OPENIN). If A=&80 it is opened for write only (OPENOUT) and if it holds &C0 the specified file is opened for random access (OPENUP). In all cases the file handle is returned in A and X and Y are left unchanged. But we said that CLOSE uses OSFIND too. This is a special case, and is



used by putting the file handle in Y and zero in A. If Y contains zero then ALL open files will be closed.

OSARGS is rather more complicated, and is used by both PTR# and EXT#. This routine requires the file handle to be in Y, and X must point to four bytes in zero page. A contains various numbers depending on the service required. If A contains 0, the value of the current pointer is returned, while if it holds 1 a new value is given to the current pointer. These are directly equivalent to !X=PTR#Y and PTR#Y=!X. If OSARGS is entered with 2 in A, the length of the file (EXT#) is returned. All the values sent or returned are held in the four bytes pointed to by X, and four bytes are needed because disc files are not limited to 64K.

Next month – advanced random files, forsaking BASIC

To complete the picture, we need operating system routines OSBPUT (address &FFD4) and OSBGET (address &FFD7), which are used by BPUT# and BGET# respectively. OSBGET is entered with the handle in Y and the value read is returned in A. ASBPUT also has the handle in Y, and the byte to be written is held in A. Now we can write our machine code version of our little Basic program.

When all has been done, RANDOM will still be open, so we'll have to close it before we can do anything else by typing

```
S%=?handle
CLOSE#S%
```

Next month we'll get onto more advanced aspects of random access files, completely forsaking Basic.

LISTING 1

```
 10 DIM Z% 200
 20 PROCass
 30 CALLstart
 40 END
100 DEFPROCass
110 OSFIND=&FFCE:OSBPUT=&FFD4:OSBGET=&FFD7:
    OSARGS=&FFDA: OSWRCH=&FFEE
120 REM OSWRCH is used to write characters to the screen
130 FOR C%=0 TO 2 STEP 2
140 P%=Z%
150 [OPT C%
160 .start
170 LDX #filename MOD 256
180 LDY #filename DIV 256
190 LDA #&80                    ; open RANDOM for writing
200 JSR OSFIND
210 TAY                         ; put handle in Y
220 LDA #0
230 JSR OSFIND                  ; close RANDOM
240 LDY #filename DIV 256
250 LDA #&C0                    ; open RANDOM for random access
260 JSR OSFIND
270 STA handle                  ; save handle
280 TAY
290 LDA #
300 STA &70                     ; set pointer to 99
310 LDA #0
320 STA &71:STA &72:STA &73
330 LDA #1:LDX #&70
350 LDX #0
360 .send
370 LDA namestring,X
380 BEQ sent
390 LDY handle
400 JSR OSBPUT                  ; send name to file
410 INX
420 BNE send
430 .sent
440 LDA #1                      ; set pointer to 99 again (information still
450 LDX #&70                    ; in zero page &70-&73)
460 JSR OSARGS
470 .read
480 JSR OSBGET
490 CMP #32                     ; read name back
500 BCC finish
510 CMP #127
520 BCS finish
530 JSR OSWRCH                  ; print each character
540 JMP read
550 .finish
560 RTS
570 .filename
580 EQUS "RANDOM"               ; EQUS and EQUB are in BASIC II only
590 EQUB 13
600 .namestring
610 EQUS "your name, etc"
620 BRK
630 .handle
640 BRK
650 ]
660 NEXT C%:ENDPROC
```

MSX

What price a new standard?

A range of MSX hardware – clockwise from top left: Hitachi, Teleton, Sanyo, Mitsubishi, Sony, Toshiba and, Centre, Cannon.

Gary Evans considers the likely impact of MSX standard computers on the UK market.

The aims of MSX are laudable – to conceive a standard specification for both the hardware and software of microcomputer systems but just how MSX machines will fair in the UK market is quite another question. Britain has in the past been dominated by machines that have exhibited a high degree of technical innovation. Most notable in this category are the string of machines marketed by Sinclair; the ZX80, 81 and Spectrum. The Commodore 64 was also, at the time of its launch, an exciting machine, being one of the first home micros to use paging techniques to provide the user with 64K of RAM. The MSX standard, by virtue of its tight definitions of the component parts of a system and of the architecture of an MSX compatible machine will mitigate against these computers showing the same sort of design flair.

It's true to say that the most notable non-MSX computer launches in recent months have adopted the safe and sure approach of MSX designs and indeed both the Amstrad CPC464 and Tatung Einstein (the machines in question) have a specification that owes a lot to the outline MSX configuration. In view of the teething problems that are often associated with machines that break new ground in terms of their design, the QL being an obvious example, the safe and sure approach has a lot to recommend it. Indeed, first time buyers will probably prefer a machine produced by a company with a reputation for producing reliable products rather than taking a chance with a product from firms with an unproven pedigree.

Another point which is very much in the favour of MSX micros is the fact that there will be a high degree of software portability. For the first time users will be able to share both commercial and home produced software.

Clone Rangers

As the heading above suggests, the fact that all MAX micros will adhere to a minimum specification, will mean that at entry point the price of many of the machines are likely to be very similar, if not in terms of appearance, certainly in respect of their performance.

The companies at present supporting British MSX hardware include Canon, Hitachi, JVC, Mitsubishi, Sanyo, Sony, Teleton and Toshiba; a list of names that at present between them dominate the UK consumer electronics industry. All of these companies were represented at a recent conference to mark the start of their various MSX marketing operations in the UK. The first of the above to make firm commitments as to delivery of their first machines were Mitsubishi and Hitachi. The Hitachi MB-H80 is a sub £200 machine that adds a feature to the basic specification. A total of 80K bytes RAM is built into the machine, 16K for the video memory, as per the basic spec. and 64K user RAM which is a distinct improvement on the 8K minimum quoted in the MSX minimum configuration. Two joystick inputs are provided along with an 8 bit parallel printer port. The processor can be clocked at any one of three switch selectable speeds, the standard calling for the 3.58MHz option. The Hitachi also makes provision for two cartridge slots, these being the main expansion route for MSX machines.

The Mitsubishi machine, designated the ML-F110, is a development of the Company's ML-8000 which has been on sale in Japan since early '83. Exact details of the UK version have not yet been announced but the major features are 32K RAM and, in common with the Hitachi machine, dual cartridge slots.

Eastern promise

All the members of the MSX group are hoping to have stocks of their machines in the shops by the autumn and so be prepared to stake a claim to the Christmas buying spree. It is likely that MSX computers will make their first appearance in the hi-fi and electrical dealers that are associated with the various existing product lines of the members of the MSX group.

It will be interesting to observe the marketing operations mounted to support MSX computers. As we have suggested, there will be little to choose between the various designs aimed at the sub £200 area of the market and it is likely that the computers will be considered largely in terms of price.

The market is going to be very crowded toward the end of this year and there are likely to be a number of bargains around as those manufacturers with existing designs

assess their position and the entry of systems such as the Amstrad CPC464 begin to have an effect.

MSX AT A GLANCE

MINIMUM SPECIFICATION

Central Processing Unit
Zilog Z80A or equivalent, running at a clock rate of 3.579545MHz.

Memory
ROM – 32Kbytes comprising Microsoft's MSX system software.
RAM – A minimum of 8Kbytes is expected. Both RAM and ROM are extendable under MSX.

Expansion Slot
Software cartridge, expansion BUS, slots.

Video Display Processor
Texas Instruments' 9918A or equivalent.

Display Modes
256 x 192 High resolution graphics.
40 x 24 Text display.
16 Colours available.

Programmable Sound Generator
General Instruments' AY-3-8910 or equivalent.
There are 3 voices available for polyphonic sound, each with an 8 octave range. A 4th noise (sound effect) channel is also available. Each channel may be modulated using one of 10 envelope waveforms.

Cassette Interface
FSK modulation. Transfer rates may be 1200 or 2400 baud.

Joysticks
One specified for the minimum specification.

The Toshiba HX10.

STANDARDISED OPTIONAL EXTENSIONS TO MSX

Screen Display
80 column text screen.

System Clock
Battery backed-up CMOS.

Communications
RS-232.

Floppy Disk
According to each company. The disk format is MS-DOS compatible.

Printer
8-bit parallel.

NEXT MONTH

Is it a better BASIC? An in-depth review of the MSX standard.

SOFTWARE REVIEWS

Caretaker

BBC/ROM/£35.00
Computer Concepts Ltd.

Caretaker is an EPROM for the BBC Micro offering toolkit services to the Basic programmer. Similar ROMs are so widely available that a newcomer to the market has to be very special to be noticed. This one is no more than run of the mill.

Certainly it has some nice functions, like the Electron-style single-key entry of a number of Basic keywords and the ability to load programs into other programs (a sort of advanced merge) and save particular sections. But the rest of the ROM is just what can be expected – no surprises.

The functions offered are *CURSOR, which turns the cursor on and off, *EXCHANGE which is a global or selective search and replace and *EXPAND which is perhaps more useful than most. It makes listings rather more readable by improving on the BBC's LIST07 and really spacing the lines out. Multi-statement lines are spread one statement per screen line, and spaces are added where necessary.

*INSERT is that clever merge mentioned earlier. You specify a filename and a line number and it does the rest. *KEYLOAD and *KEYSAVE verge on the fatuous – they simply load and save the function key buffer. *LOAD and *SAVE are perfectly adequate so there is really no need for these commands.

*LVAR prints the values of any predefined group of Basic variables. This command is possibly a little more comprehensive than it needs to be, but it is nice nonetheless. *MERGE joins a program on file (disc or tape) to a program already in memory. *MOVE moves a program to a new specified value of PAGE.

*NOTAB sets the TAB key to act as normal, while *TABSTOPS causes it to tabulate the cursor by defined amounts. *PARTSAVE allows you to specify which part of a program you want to save, and *RENUMBER is a very enhanced version of the BBC's facility, allowing blocks to be renumbered and moved whilst leaving the rest of the program as it is. *RETRIEVE is another of those routines to help you get over the annoying 'Bad program' error and works as well as any other. *SQUASH selectively removes spaces and REMS in a program, as well as joining lines together if you so choose. Finally *STATUS prints the values of the relevant Basic system variables and the amount of memory free.

CARETAKER is essentially worth the money if that's what you want, but if offers nothing radically new or stunningly useful, joining the host of other ROMs in the reviewers' dust-gathering ROM board.

SOFTWARE FILE

A BUYER'S GUIDE TO UTILITY SOFTWARE

This month software file looks at packages which assist the artistic side of computing: graphics packages, sound generators, printer and screen dumps, and character generators – games writers look no further!

In August we'll have the lowdown on the many programming languages available for home micros.

MACHINE	PRODUCT	PRICE	FORMAT	MEMORY	SUPPLIER	COMMENT
CHARACTER GENERATORS						
BBC	SHAPE GENERATOR	10.06	C	32K	MOLIMERK	ENABLES SHAPES TO BE CREATED ON GRID
BBC	THE GENERATORS	6.95	C	32K	QUICKSILVA	CHARACTER AND TELETEXT GENERATOR WITH INSTRUCTIONS
BBC	UTIL-1	9.95	C	32K	GEM	CHARACTER DEFINE AND ENVELOPE EDITOR
BBC	DEFINE	15.53	C	32K	MOLIMERK	CUSTOM DESIGNED CHARACTERS ON 16 x 16 GRID
BBC	MAGIC WINDOW	8.95	C	16K	QUICKSILVA	JOYSTICK CONTROL
BBC	SPRITE AID	6.95	C/D	UX	ADAMSOFT	JOYSTICK OR KEYBOARD CONTROL
BBC	SPRITES	NA	C/D	UX	BEEBUG	HIGH SPEED ARCADE GAMES CAN BE WRITTEN IN BASIC
CBM64	SUPERFONT	6.95	C	64K	ENGLISH SOFTWARE	REDEFINE AND SHAPE CHARACTER GENERATOR
CBM64	SPRITELIB	6.00	C	UX	PICOSYSTEMS	EDITOR AND SPRITE LIBRARY
CBM64	SPRITMAKER	6.95	C	64K	ENGLISH SOFTWARE	FULL EDITOR SPRITE GENERATOR
DRAGON	CHARACTER GENERATOR	9.95	C	32K	GEM	CREATION AND RECALL OF CHARACTER SETS
DRAGON	PIXEL EDITOR	10.95	C	32K	DRAGON DATA	EACH PIXEL CAN BE ACCESSED
ORIC	CHARACTER DESIGN	6.95	C	48K	TOMORROWS WORLD	FULL DISPLAY AND SPRITE LIBRARY
SPECTRUM	SPECTRUM SPRITES	5.00	C	48K	WORK FORCE	SPRITE GENERATOR
SPECTRUM	DISPLAY	7.00*	C	16K	WORK FORCE	ALLOWS USER-DEFINABLE GRAPHICS
VIC 20	PIXEL POWER	7.95	C	8K	QUICKSILVA	CHARACTER GENERATOR
GRAPHICS						
ATARI	GRAPHIT	14.99	CR	16K	ATARI	PLOTS PIE OR BAR CHARTS/3D DISPLAYS
ATARI	PILOT	5.99	CR	8K	ATARI	TURTLE GRAPHICS WITH SOUND STATEMENTS
BBC	ARTFUN	9.95	C/D	32K	RH ELECTRONICS	RH LIGHTPEN SOFTWARE
BBC	ARTIST 2	10.00	C	32K	BEEBUG	MODE 2 DRAWING SYSTEM (REQUIRES JOYSTICK)
BBC	BEEBART	14.95	C	32K	QUICKSILVA	JOYSTICK COMPATIBLE PAINTING SYSTEM
BBC	CREATIVE GRAPHICS	9.95	C	16K	ACORNSOFT	36 PROGRAMS INC. ANIMATION AND 3D
BBC	DESIGN	10.00 19.00	C D		BEEBUG	GRAPH AND CHART PROCESSOR WITH USER DEFINED CH'TERS FOR TECH DRAWING
BBC	EDG GRAPHICS	19.95	C	32K	SALAMANDER	CAPABLE OF BUILDING COMPLEX DRAWINGS AND DESIGNS
BBC	EDG GRAPHICS 2	24.95	C	32K	SALAMANDER	EXTENDED VERSION OF ABOVE WITH AUTO-TRANSFER TO DISC
BBC	GRAPHS+CHARTS	9.95	C	32K	ACORNSOFT	GRAPH ROUTINES FOR INCORPORATION WITHIN USER'S PROGRAMS
BBC	GRAFDISK	12.95	D	32K	CLARES	'LOW-COST CAD'
BBC	GRAFKEY	7.95	C	32K	CLARES	CASSETTE VERSION OF ABOVE
BBC	GRAPHICS PACK	9.50	C	32K	BUG-BYTE	ELASTIC BANDING, ZOOM, 3D
BBC	GRAPH PLOTTER	16.10 19.95	C C	32K	MOLIMERX	MATHEMATICAL GRAPH PLOTTER WITH EQUATION BASE DATA
BBC	GRAPHICS	33.25	R		COMPUTER CONCEPTS	SPRITE GRAPHICS/LOGO TURTLE GRAPHICS/ GEN. PURPOSE GRAPHICS COMMANDS
BBC	MAGIC WAND	10.06 13.51	C D	32K	MOLIMERX	SHADING/ROTATIONS/TRANSFORMATIONS/ REFLECTIONS/SCREEN DUMP
BBC	MCVID	14.95 15.95	C D	32K	PICA	MC ROUTINES – WIRE DRAWING/SPRITE DRAW/SCROLLING IN ALL MODES
BBC	PAINTBOX	10.00 12.00	C D		BEEBUG	JOYSTICK REQUIRED. VERY COMPREHENSIVE DRAWING PACKAGE – LOW-COST CAD
BBC	PICTURE MAKER	9.95	C	32K	ACORNSOFT	ROTATION/COLOUR TRANSFER/RESCALING ARCS/CIRCLES/TRIANGLES
BBC	SUPERPLOT	10.00	C	32K	BEEBUG	SCREEN REPRESENTATION OF MATHS FUNCTIONS
BBC	TELETEXT PACK	10.00 12.00	C D	32K	BEEBUG	EDITS BBC MODE 7 SCREEN
BBC	TURTLE GRAPHICS	16.85 19.90	C D	32K	ACORNSOFT	IMPLEMENTATION OF LOGO SUBSET
CBM 64	KOALA PAINTER	79.95	D	UX	AUDIOGENIC	PACKAGE INCLUDES GRAPHICS TABLET
CBM 64	GRAPHIX 64	10.00 12.00	C D	64K	SUPERSOFT	PLOT/DRAW/FILL/DISPLAY
CBM 64	HIRES GRAPHICS	15.00	C	64K	SUE'S SOFT	12 BASIC COMMANDS INC. SPLIT SCREEN
CBM 64	SCREEN GRAPHICS	12.95 14.95	C D	UX	ADAMSOFT	24 NEW BASIC COMMANDS FOR HI-RES GRAPHICS AND SPRITE GENERATION

MACHINE	PRODUCT	PRICE	FORMAT	MEMORY	SUPPLIER	COMMENT
CBM 64	SPRITE EDITOR	5.00	C	UX	STACK	SELF EXPLANATORY
CBM 64	SPRITE MAKER	7.00	C	UX	ENGLISH	
CBM 64	SPRITEMASTER	7.00	C	UX	WESSEXSOFT	
DRAGON	ARTIST'S DESIGNER	6.95	C	32K	WINTERSOFT	TEXT MERGING/SHAPE/STORE ETC.
DRAGON	DOODLES	5.00	C	16K	AUTOMATA	DEMONSTRATION PROGRAM WITH USER DEFINED CHARACTERS
DRAGON	ANIMATOR	4.95	C	32K	DRAGON DATA	JOYSTICK CONTROL OF GRAPHICS CREATION BY FLIPPING THRO' PAGES
DRAGON	GRAPHICS SYSTEM	9.95	C	32K	SALAMANDER	JOYSTICK REQUIRED/DRAW PICTURES TEXT AND SAVE
DRAGON	SKETCHPAD	4.65	C	32K	MORRISON	MC DRAWING AID
ORIC	ORIC CAD	9.99	C	48K	TANSOFT	ENLARGE/FILL/ROTATE A DISPLAY OBJECT
SPECTRUM	DOODLES	5.00	C	16K	AUTOMATA	AS FOR DRAGON VERSION (ABOVE)
SPECTRUM	DYNAMIC GRAPHICS	14.95	C	48K	PROCOM	'PROFESSIONAL' GRAPHICS DESIGN AID
SPECTRUM	MELBOURNE DRAW	8.95	C	48K	MELBOURNE HOUSE	PARTICULARLY SUITED TO GAMES WRITERS
SPECTRUM	MULTIGRAPHICS	6.90	C	16K	BRIDGE SOFT	MENU DRIVEN GRAPHICS TOOLKIT
SPECTRUM	PIXEL PICTURES/ DISPLAY PICTURES	7.50	C	48K	BUGSOFT	ATTRIBUTES/BORDER/DRAW COMMANDS GRAPHICS STORE AND LOGO UTILITY
SPECTRUM	SPECTRO GRAPHICS	6.90	C	48K	BRIDGE	ELEVEN GRAPHICS AND TEXT PROCEDURES
SPECTRUM	VIEWPOINT	6.50	C	16K	ACS	3D GRAPHICS USING BASIC ROUTINES
VIC 20	GRAPHVICS	9.95 11.95	C D	3K	ADAMSOFT	18 NEW BASIC COMMANDS
VIC 20	HIRES/MULTICOLOUR	8.95 10.95	C D	UX	ADAMSOFT	14 COMMANDS FOR PLOTTING
VIC 20	JOYSTICK PAINTER	6.95 8.95	C D	3K	ADAMSOFT	JOYSTICK REQUIRED – COLOUR CHANGE/ ERASE/DRAW IN 3 COLOURS
ZX81	GRAPHICS STARTER	4.95	C	1K	BRIDGE	EXPLAINS ZX81 GRAPHICS PROCEDURES
ZX81	GRAPHICS TOOLKIT	4.95	C	16K	JRS	MC ROUTINES FOR INCLUSION IN USER PROGRAMS
ZX81	HIRES	5.95	C	16K	COMP. RENTALS	8 COMMANDS FOR HI-RES ON THE ZX81
ZX81	MULTIGRAPHICS	6.90	C	16K	BRIDGE	PROCEDURES TO CONTROL ZX81 GRAPHICS FUNCTIONS
ZX81	SCREEN ANIMATOR	5.00	C	16K	MICRO-SIGN	62 INTERACTIVE 16x12 SCREENS CAN BE COMPOSED AND SAVED
ZX81	SCREEN KIT 1	6.00	C	4K	PICTURESQUE	ADDITIONAL GRAPHICS COMMANDS
XZ81	VIEWPOINT	6.50	C	16K	ACS	3D GRAPHICS ROUTINES
PRINTER SCREEN UTILITIES						
BBC	PRINTMASTER	33.95	R	UX	COMPUTER CONCEPTS	SUPPORTS RANGE OF EPSON DEVICES
BBC	MAXIGRAPH	16.10	D	32K	MOLIMERX	GRAPHICS PRINTING COMMANDS
BBC	HIRES	8.00	D	32K	MUSE	SCREEN DUMP UTILITY
CBM64	CENTRONICS	8.00	C/R	5K	AUDIOGENIC	PROTOCOLS FOR CENTRONICS PRINTER
DRAGON	MODE 5	6.95	C	16K	SOFTEK	TEXT AND GRAPHICS MIXER
DRAGON	HIRES	25.30	CR	32K	COMPUSENSE	51 x 24 SCREEN FORMAT
DRAGON	HIPRINT	7.95	C/D	32K	PREMIER	DUMPS HI-RES DISPLAY
SPECTRUM	TASWIDE	5.50	C	16K	TASMAN	64 CHARACTER/LINE DISPLAY
SPECTRUM	MICRO-PRINT	5.00	C	UX	MYRMIDON	SCREEN PRINT DUMP OF 24 LINES x 42/51 CHARACTERS
SPECTRUM	SYS64	6.95	C	16K	ARTIC	64 CHARACTERS PER LINE
SPECTRUM	HI-T	5.95	C	16K	TIMEDATA	SCREEN ENCHANCER WITH CHARACTER SET, WINDOWS ETC.
VIC 20	CENTRONICS	7.95	C	UX	AUDIOGENIC	PROGRAM CAN BE LOADED UNDERNEATH AND PRINTED OUT OF PRINTER
VIC 20	SUPER EXPANDER	6.95	C/D	8K	ADAMSOFT	COPIES HI-RES GRAPHIC SCREEN
SOUND						
BBC	BEEBSYNTH	7.95	C/D	32K	CLARES	ENVELOPE DEFINING SYNTHESISER REQUIRES 1.2 OS IN DISC
ORIC	SYNTHESIZER	7.95	C	48K	TOMORROWS WORLD	VARIABLE SPEED AND PITCH WITH SAVE OPTION
SPECTRUM	WHITE NOISE	5.95	C	16K	GILSOFT	30 NEW COMMANDS TO DEFINE SOUND AND GRAPHICS

Gremlin

BBC/ROM/£35.00
Computer Concepts Ltd.

Computer Concepts seem to be doing strange things these days. Once they were pioneers in BBC sideways ROMs, now they're following the crowd. Or so it seems when you take a look at Gremlin and the elsewhere-reviewed Caretaker EPROMs.

Gremlin is yet another machine code monitor and has been on Computer Concepts' lists for a long time, although only hitting the streets comparitively recently. It is probably one of the least useful of all the monitor ROMs one can buy. Although it offers the standard monitor facilities of memory move, memory amendment, register examine and adjust, even down as far as the immensely useful single step function, it makes these simple things so awkward to use that one gets the feeling the author has never used a reasonable monitor before.

Its command structure ostensibly borrows a lot from that systems programmers' favourite language – C – but don't believe everything you read in manufacturer's manuals. C was never this bad. Each register and 'system variable' is actually assigned its own variable within Gremlin which totally precludes the STANDARD and widely accepted practice of altering registers and memory. Breaking new ground it may be, but this ground should be left covered up!

Then there's the ridiculous system of switches to turn various functions on and off, like single step. Why on earth would you want to be able to turn THAT off? These switches extend to such meaningless things as formats and the possible ways of entering hex numbers. The manual tells us that this is so that Gremlin's disassemblies can be sent to file (which IS useful) and then *EXECd from Basic, but machine code on a 6502 is unfriendly enough without the added futility of a programmer's whim.

The only thing one can say about Gremlin in conclusion is that it may be cheap, but it's ghastly. Come on Computer Concepts, there's still time before we forget masterpieces like the graphics ROM, Printmaster and Wordwise. Plug yourselves in. **AD**

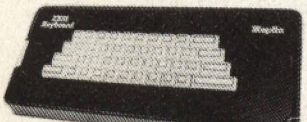
SUPPLIERS SUPPLIERS SUPPLIERS SUPPLIERS

Acornsoft	0223 316039
ACS	0532 667440
Adamsoft	0706 524304
Addison Wesley	01 631 1636
Amersham Software	02403 6231
Argus	01 437 0626
Artic	0401 43553
Ashby Computer Centre	
Audiogenic	0734 586334
Automata	0705 735 242
Aztec	0742 862246
Beebugsoft	0727 54280
Bridge Soft	061 427 6107
Bugbyte	051 7097071
CBM	0753 79292
Clares	0606 48511
Compusense	01 882 0681
Computer Concepts	
Computer Rentals	01 533 2918
CP Software	02406 3783
Crystal	
Dragon Data	0656 744700
Durell	0823 54489
Dynatech	0481 20155
ECCE Productions	01 302 1667
East London Robotics	01 474 4430
English Software	061 835 1358
Gem	0279 723518
Gilsoft	
Golem	0344 50720
J K Gosden	
Hilton	0689 35101
Hisoft	0296 688995
Impex	01 900 0999
JRS	0903 65691
Kotra	0491 572512
Kuma	07357 4335
Lerm	
Level	021 643 6728
Lothlorian	0625 876642
Malva	
McGraw Hill	0628 23432
Melbourne House	044 282 2930
Micro-Aid	0209 831 274
Micro?Sign	04862 67847
Micropower	0532 683 188
Molimerx	0424 223 636
Morrison	0532 480987
Myrmidon	073784 2072
Nectarine	
Oxford Computing	0753 888866
Peak	0283 44904
Pico	04022 24595
Picturesque	01 777 0372
Premier Microsystems	01 659 7131
Procom	01 508 1216
PSS	
Quicksilva	0703 20169
RH Electronics	0223 311 290
Salamander	0273 771 942
Scimitar	
Severn	0594 43352
Sinclair	0276 685311
Softguard	
Softek	01 240 1422
Stack	051 933 5511
Supersoft	01 861 1166
Tansoft	02205 2261
Tasman	0532 438301
Timedata	
Tomorrows World	06845 62467
Vision	01 748 9009
Wintersoft	01 367 5720
Workforce	

EXCITING ADDITIONS FOR YOUR HOME COMPUTER

KEYBOARD with ELECTRONICS for ZX SPECTRUM

★ Full size, full travel keyboard that simply plugs into expansion port on your Spectrum. ★ Offers single key selection of all major multi-key functions. ★ Extends port for other peripherals. ★ Can accept Atari-type joysticks (optional extra — order 2 of FG66W, £1.36 each and note that case will require cutting).
Three kits needed to build unit: Order LK29G, LK30H & XG35Q. Total price £39.95. Full construction details in Project Book 9 XA09K 70p. Also available ready-built. Order As XG36P. Price £44.95.



KEYBOARD with ELECTRONICS for ZX81

★ Full size, full travel keyboard that's easy to add to your ZX81. ★ No soldering in ZX81; simple instructions make it easy to fit. ★ Makes Shift Lock, Function & Graphics 2 single key selections.
Complete kit (excl. case) LW72P Price £23.95. Case XG17T £4.95. Full construction details in Project Book 3 XA03D. Price 70p. Ready-built in case XG22Y. Price £32.50.

MODEM

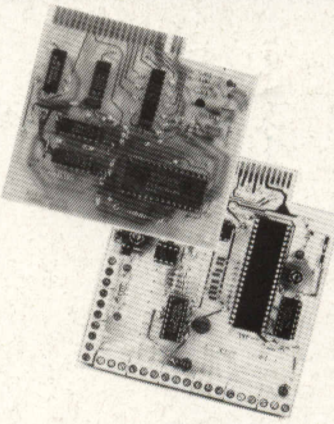
A CCITT standard modem that connects directly to your telephone line via a BT approved transformer. Transmits and receives simultaneously on European standard frequencies at 300 baud. May be used to talk to any other 300 baud European standard modem including the Maplin Computer Shopping modem on 0702 552941 and any British Telecom Datel 200/300 Service modem. The modem's computer interface is RS232 compatible.
Complete kit (excl. case) LW99H. Price £44.95. Case YK62S £9.95. Full construction details in Project Book 5 XA05F Price 70p.

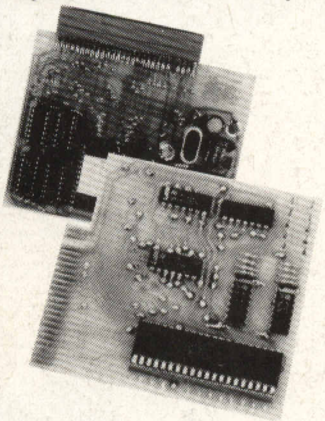
INTERFACES for MODEM

Interfaces are now available for the following machines: Commodore 64, Dragon 32, Oric, Spectrum, VIC20 and ZX81. Each is complete with a Machine Code Communications program. The BBC micro needs no interface and a suitable program is on Maplin catalogue page 15 or Project Book 8, page 59.

Computer	Order	Details	Price
64/VIC20	LK11M	Book 7	£9.45
Dragon 32	LK12N	Book 8	£14.95
Oric 1	LK40T	Book 10	£13.95
Spectrum	LK21X	Book 8	£19.95
ZX81	LK08J	Book 7	£29.95

**Project Book 7 XA07H. Price 70p.
Project Book 8 XA08J. Price 70p.
Project Book 10. XA10L. Price 70p.**





ZX81 I/O PORT

★ Provides two bi-directional ports for 16 input or output lines. ★ One buffered output which can interface directly to CMOS. ★ On board address selection permits expansion to six ports with two boards.
Complete kit LW76H. Price £10.49. Full construction details in Project Book 4 XA04E. Price 70p.

MAPLIN TALK-BACK SPEECH SYNTHESISERS

★ Unlimited vocabulary with allophone (extended phoneme) system. ★ Can be used with unexpanded Oric 1, VIC20 or ZX81 as it does not require large areas of memory. ★ Speech may be easily added to programs. ★In VIC20 version speech output is direct to TV speaker with no additional amplification needed.

Computer	Order	Details	Price
Oric 1	LK28F	Book 9	£23.95
VIC20	LK00A	Book 6	£22.95
ZX81	LK01B	Book 6	£19.95

**Project Book 6 XA06G. Price 70p.
Project Book 9 XA09K. Price 70p.**

DRAGON 32 I/O PORT

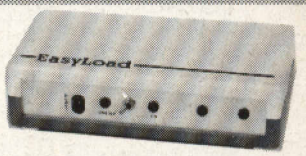
★ Provides two TTL & 3-state bus compatible 8-bit ports, ★ Four norm/inv. latched ports, ★ Two relay switched ports ★ And two opto switched ports. ★ Module plugs directly into cartridge socket and is fully programmable from BASIC.
Complete kit LK18U. Price £14.95. Full construction details in Project Book 8 XA08J. Price 70p.

MAPLIN CATALOGUE

Full details of all Maplin's projects and electronic components in our huge 480 page catalogue. On sale now in all branches of W.H. Smith price £1.35. Or send £1.65 (incl. p&p) to our mail order address.

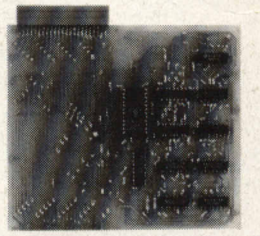
OTHER PROJECTS

For full details of our other computer-related projects please see the relevant project book as below:
**BBC Motherboard — Book 11.
ZX81 Sound on TV — Book 6.
ZX81 Extendi-RAM — Book 9.
VIC20 Extendiboard — Book 9.
Dragon 32 Extendiport — Book 10.
TTL/RS232 Interface — Book 9.
Project Book 6 XA06G. Price 70p.
Project Book 9 XA09K. Price 70p.
Project Book 10 XA10L. Price 70p.
Project Book 11 XA11M. Price 70p.**



SPECTRUM EASYLOAD

★ Greatly reduces cassette LOADing & SAVEing problems on Spectrum. ★ Battery powered, no bus connections. ★ Charging from Spectrum PSU. ★ SAVE & LOAD indicators.
Complete kit (excl case) LK39N. Price £9.95. Full construction details in Project Book 10 XA10L. Price 70p.

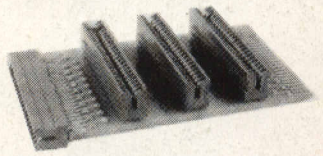


ZX81 HI-RES GRAPHICS

★ Full 256 x 192 fine pixel display with normal or inverted video. ★ Draws lines, circles and triangles, fills and textures. ★ Up to 32 user defined graphics. ★ Operates directly from extended BASIC.
Complete kit LK23A. Price £27.50. Full construction details in Project Book 9 XA09K. Price 70p.

ZX81 SOUNDS GENERATOR

★ Turns your ZX81 into a mini-synthesiser. ★ 3 programmable tone generators. ★ 3 programmable attenuators. ★ Noise generator with 3 pitch levels for special effect sounds. ★ Single address access with PEEK & POKE. ★ Connects directly to extension board or expansion port socket with extra socket (order RK35Q £2.20) ★ Requires separate amp and speaker.
Complete kit LW96E. Price £13.49. Full construction details in Project Book 5 XA05F. Price 70p.



ZX81 EXTENSION BOARD

★Plugs directly into ZX81 expansion port. ★ Accepts a 16K RAM pack and three other plug-in modules simultaneously. Parts are sold separately as follows:
PCB GB08J. Price £2.40. Edge Connectors (4 needed) RK35Q. Price £2.20 each. Track pins (1 pack needed) FL82D. Price 85p per pack of 50.

MAPLIN
ELECTRONIC SUPPLIES LTD

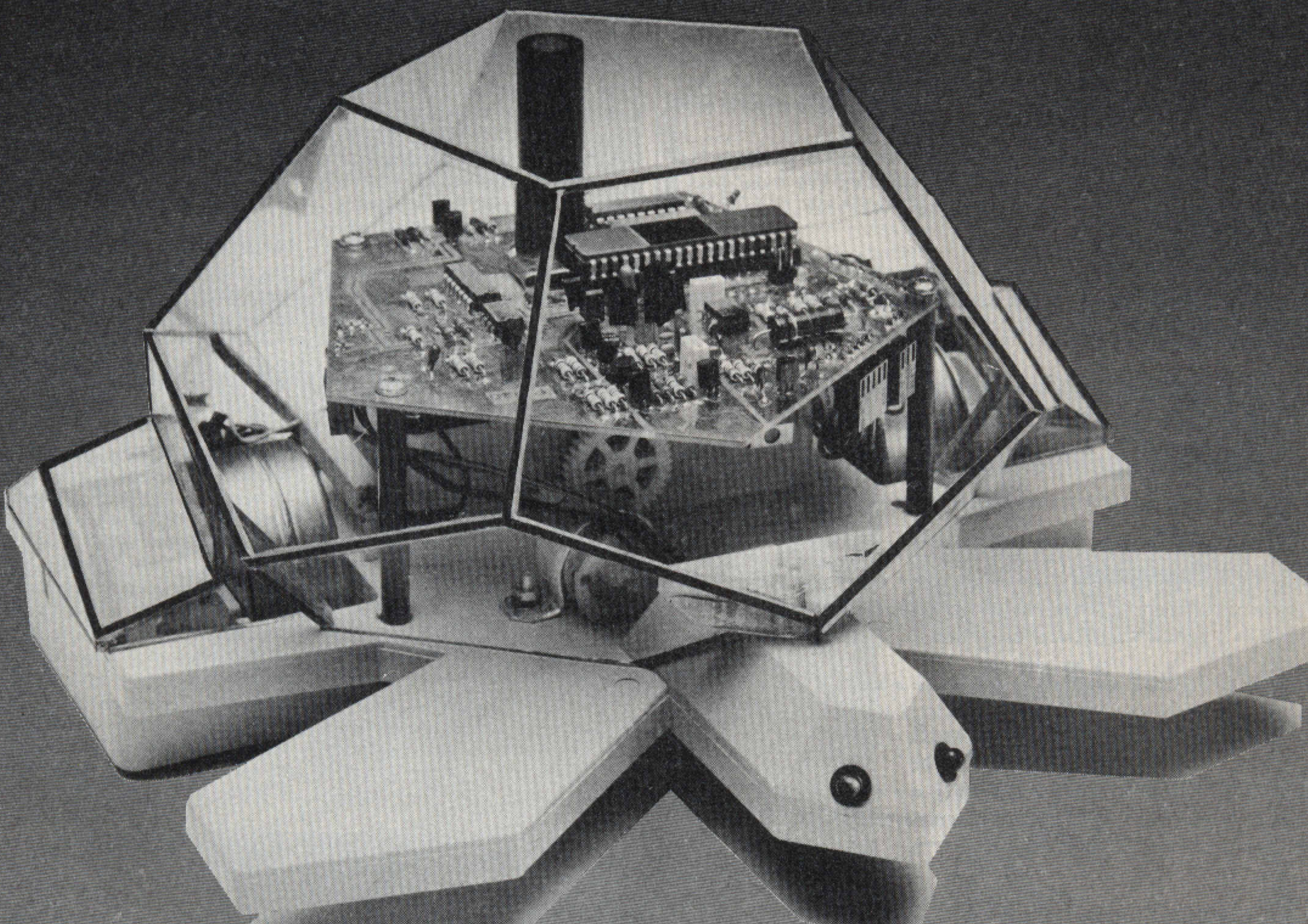
**Maplin Electronic Supplies Ltd. Mail Order: P.O. Box 3, Rayleigh, Essex SS6 8LR.
Tel: Southend (0702) 552911. • Shops at: 159-161 King Street, Hammersmith, London W6. Tel: 01-748- 0926.
• 8 Oxford Road, Manchester. Tel: 061-236-0281. • Lynton Square, Perry Bar, Birmingham. Tel: 021-356-7292.
• 282-284 London Road, Westcliff-on-Sea, Essex. Tel: 0702 554000. • 46-48 Bevois Valley Road, Southampton. Tel:0703 25831. All shops closed all day Monday.
All prices include VAT and carriage. Please add 50p handling charge to orders under £5 total (except catalogue).**

your ROBOT

An EMAP Publication

BRITAIN'S FIRST ROBOTICS MAGAZINE JULY 1984

INFRA-RED TURTLE
The Valiant previewed



NEW SERIES - CONSTRUCTING LEGO ROBOTS

THE ROBOT CONGRESS - a report from albuquerque

EDITORIAL

VOLUME 1 ISSUE 6

While talk about the theory of robots is a worthwhile pursuit, it's not until more people begin building systems and turning their minds to the uses to which they can be put, that the art of robotics will begin to progress. To this end, we begin a series of articles that will show you how, for very little outlay, robots of surprising sophistication can be built.

The method of construction chosen means that our DIY robots can easily be modified to suit individual needs and applications. We hope that a great many of you will be tempted to tackle construction of the devices and look forward to hearing of your exploits in the future.

THE US SCENE

*Elsewhere in this issue of **Your Robot** Peter Matthews reports on his visit to a recent Robotics exhibition held in the US. We'll leave Peter to give the full story but one aspect of the report is very interesting. This is the fact that at present the American hobby and low cost robot scene is at about the same stage of development as our own involvement in this field. Indeed some would say that we are ahead of the Americans in robotics.*

It's to be hoped that we can maintain this lead as all aspects of robotics are likely to pay an increasingly important role in our lives and it would be nice to think that once ahead, we in the UK can stay ahead.

CONTENTS

Editorial 01-833-0846
Editor Gary Evans
Assistant Editor William Owen
Production Editor Liz Gregory
Contributing Editors
Gary Herman, Peter Matthews
Advertising 01-833-0531
Advertisement Manager Richard Jansz
Production 01-833-0531
Art Editor Jeremy Webb
Make-up Time Graphics
YOUR ROBOT
2nd FLOOR
155 FARRINGDON ROAD, LONDON EC1R 3AD

NEWS

THE VALIANT TURTLE

The new remote-controlled Valiant turtle is an extremely attractive teaching tool which may be interfaced with virtually any micro. Aimed specifically at the education market, the robot is provided as part of a package which comes ready to run upon purchase.

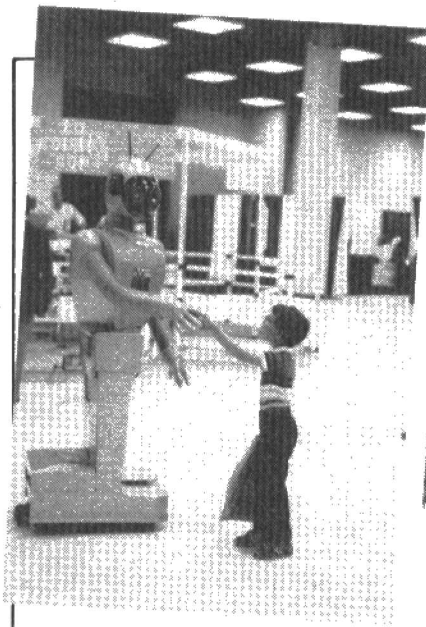
The turtle is certainly rather futuristic to look at and, with its perspex shell, should prove to be very popular with primary school children. The see-through shell allows the user to watch the motors actually working as the turtle moves around. It will be moulded when the model is in full production but at the moment looks a little angular and awkward at prototype stage. It remains to be seen just how the design will stand up to classroom enthusiasm.

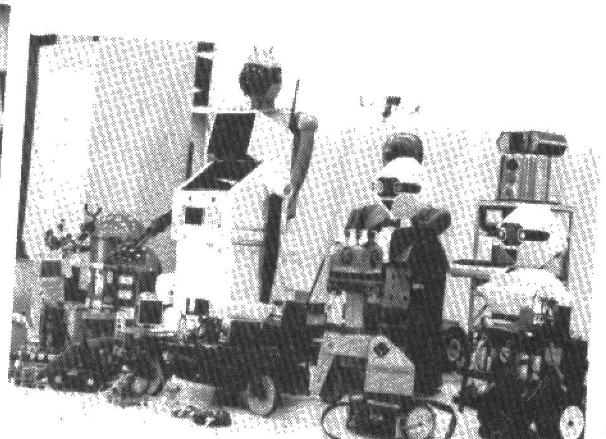
The infra-red control will allow operation of the turtle at distances of up to six metres and does not enjoy working at ranges of less than 6 feet. The advantages of remote control are obvious and the increased range of movement will allow this turtle to be rather more mobile than 'plugged in' varieties.

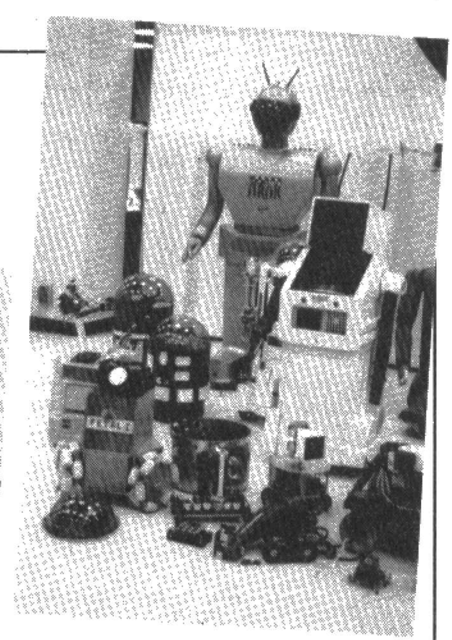
The Valiant is operated by a Logo program which enables it to move in stages as tiny as 1mm and turn through angles of 1° at a time. It holds a Berol pen of any thickness which can be changed without opening up the inside of the turtle. This should prove advantageous as different colours and line thicknesses may be produced without much trouble. Powered by a rechargeable battery, the turtle will draw continuously for at least two hours and when it does need recharging, the glow in its eyes will fade.

The whole Valiant package consists of an interface disk for the relevant computer, an infra-red control and connector, a plug adaptor and a manual. Added to this, a magazine is provided with suggestions for use of the turtle and by way of assistance for those unfamiliar with micros. Aptly named, PENUP seems to be a nice addition to the package as a whole.

*The full production models of the Valiant will be available from June onwards. The retail price is £199.99 plus VAT but it will be available for educational establishments at £150. Prototype models are already being used in some schools and in a future issue of **Your Robot** we will be reporting upon how the Valiant stands up to a school environment.*







THE U.S. DIRECTION

Peter Matthews has been examining the state of the art in U.S. robotics. This is his first report, from the International Personal Robot Congress, Albuquerque, New Mexico.

The International Personal Robot Congress at Albuquerque in New Mexico was a show piece for the American small robot scene. The only notable absence was Microbot, the first and probably best arm type robot which first appeared a few years ago. The Congress was then reviewing the achievements of about two years of work and development for the American designer and manufacturer of small robots. Inevitably, the comparison has to be made with British achievement, and to my surprise, the USA and its technology offered no more than our own. In fact, it turned out that the people who run the industry and what they had to say about the future, were more interesting than the exhibits.

The visitors to the show were probably more computer literate than they would have been in some other parts of middle America. It is fairly certain, however, that there would have been more 'buzz' to the show if it had been held in Los Angeles or New York. Nevertheless, the number of visitors was reasonable and their technically critical knowledge gave challenge to the displays and their robots. Although the exhibition hall was medium-sized by American standards, the people who served in various capacities in the companies in the micro robotics industry proved to be very interesting.

The show was in two parts, the professional and the amateur section with the former being the best part. A two day congress with a battery of speakers from the USA and Europe was a large part of the show. They talked on all subjects including hardware, software, business opportunities, venture capital, applications and robots in the future. The talks were all professionally recorded and we now have almost thirty cassettes on matters robotic. Much of this will appear in articles, or at least commentary in the magazine over the next few months.

The British contingent were quite well represented. Robin Bradbeer and David Buckley, both journalists and designers, were evident as well as our own David Moyel and of course the ubiquitous Doctor Billingsley. The Cybermate robot was there and this gave John Billingsley the opportunity to show the American operators how British roboticists sort out the bugs in an operating system.

LECTURES

One of the chief speakers who opened the conference was Isaac Asamov the famous science fiction writer. Such books as 'I Robot' with the now well known laws of the robot have changed some public attitudes to robots. This made him a natural opening speaker for the conference. The great man did not appear in person but by a telegraphic link which projected him live onto a large screen. He was able to hear and answer questions and even had a slightly querulous exchange with a 'crossed line' to the amusement of the audience. During his talk we learned the reason for his not coming to the congress. Apparently he has no faith in transport which have a high technology content and will not trust himself to them (since when have trains been hi-tech?). You could almost hear the audience thinking "If he, as a leading technologist, does not trust trains and planes what does he know that we do not and should we cancel our flights home!"

Joe Engleburger was also a star speaker. As pioneer of the industrial robot he has taken a growing interest in the smaller variety over the last couple of years. His support and interest for many small companies in micro robotics has been something that many of us remember with pleasure. Then followed the sessions of talks, discussions and workshops with speakers from all levels of interest in microrobots. Doug Bonham, Chief Executive of Heathkit who produce Hero, Skip Stevely of Androbot and many others described their experiences in the robot marketplace during the last few years. Representatives of other companies such as Gillette, Exxon, Westinghouse, Xerox and many others questioned the speakers closely about the present and future estimates of the market for small robots. Not all of them will be entering onto the market but most of the big companies are observing its size and promise pretty closely.

An essential part of any jamboree of this kind is a contest and prizegiving. The obvious one was, of course, a competition for the best design of a robot and this attracted about a dozen entrants. They ranged from a robot lawnmower to a robot balloon. The whole thing was a great deal of fun and the prizewinners were given a range of prizes on the concluding day.

The Congress was enormous fun although it didn't really have the size and scope of an international event. It tended to raise more questions about small robots than it gave answers but then that is always the case with a young and growing technology. There were some answers however, and many of them interesting. We will tell you about some of these and the products, markets, applications and personalities at the show over the new few issues of **Your Robot.**

MOTORS EXPLAINED

D. S. King continues his assessment of the different types of motors which can be used for control applications.

Since a stepper rotates in fixed incremental steps, that is in direct response to electrical pulses, it is directly compatible with digital inputs from a computer. **Figure 1** shows the simple arrangement for motor inputs, and we shall later look closer at the way pulses are tailored so as to be acceptable to the stepper.

There are two basic ways of connecting a supply to a 4-phase stepper. In mode-1, a direct connection is made to the supply. But there is an advantage in including series resistors, as shown in mode-2, for it allows higher voltages to be used and thus gives faster stepping rates.

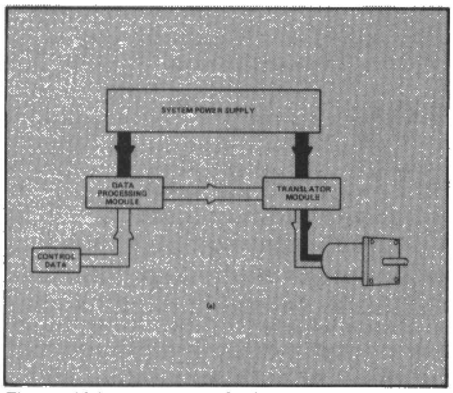


Figure 1(a) shows a typical stepper motor control and drive configuration.

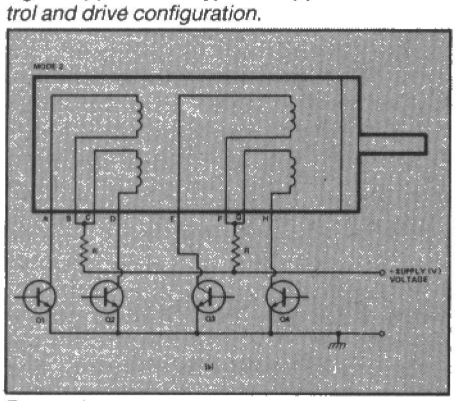


Figure 1(b) depicts a 4-phase stepper with series resistors which allow for faster stepping rates.

Although stepper motor designs vary, it is the permanent-magnet (PM) type, **Figure 2,** that has most interest for computer controlled equipment because of its particular set of characteristics and relatively low cost. **Table 1** sets out their characteristics and compares them against those of the dc motor.

When the pulse train from a stepper is of the order of 200 steps/s, and if the motor's load and inertia permit, it is possible to rotate the motor in either direction, as well as stopping, starting and reversing in instantaneously. Consequently, the stepper is ideal for positioning, providing stepping movements are acceptable to the application.

Steppers can drive floppy discs and are

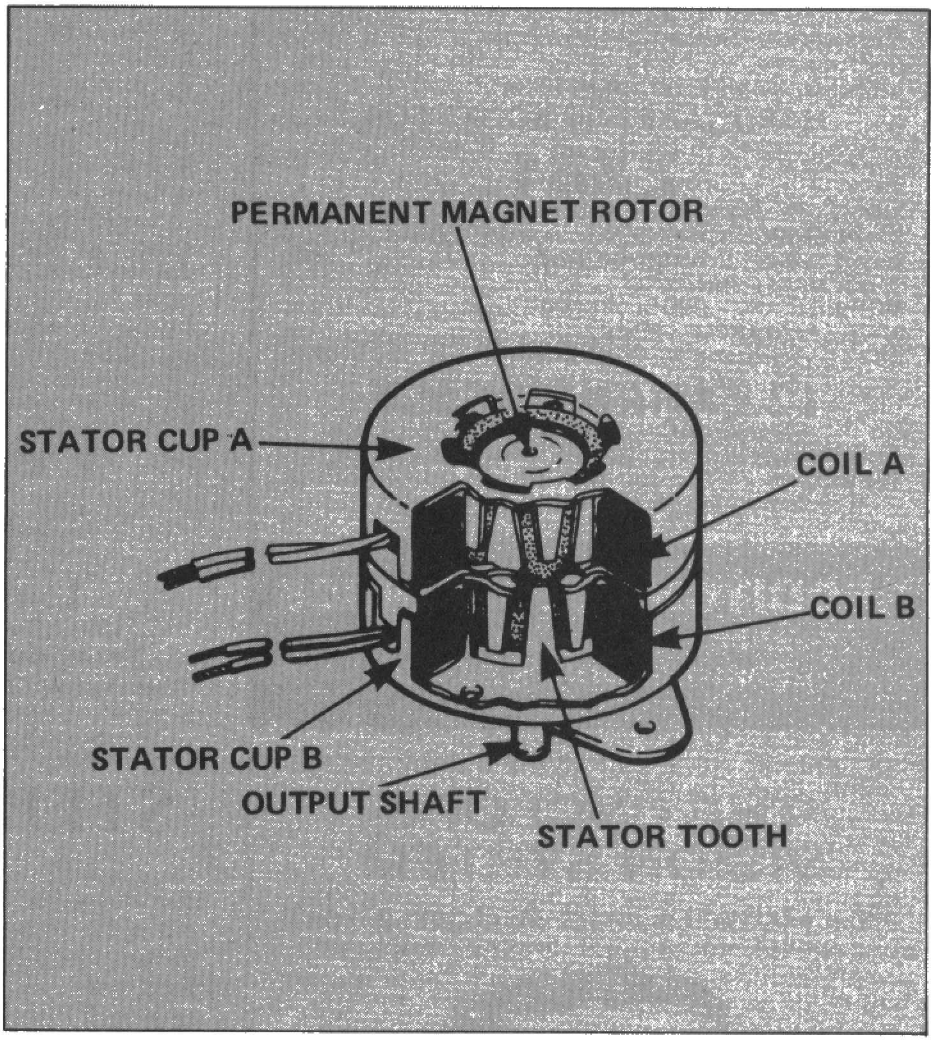


Figure 2 looks at the construction of a 2-phase stepper.

TABLE 1. PM motor characteristics (stepper and D.C. motors)

Stepper motor

* ★ Accurate positional knowledge without use of expensive shaft encoding.
* ★ Accurate open loop speed control. Cogging effect could be undesirable at low speeds for some applications.
* ★ Efficiency is relatively low in terms of electrical/mechanical power transmission.
* ★ Sensitive to high inertial loads because of stepping mode. Electronic pulse ramping often needed for acceleration/deceleration in high-inertia/high-speed applications.
* ★ Stepper has high holding torques both energinsed and unenergised.

D.C. motors

* ★ Iron-rotor.
  Torque/current nearly linear.
  Good start-up torque.
  Medium efficiency.
  Relatively high current rating because no field I^2R losses.
* ★ Ironless-rotor.
  High power-to-volume.
  Low rotor inertia. Fast response to signals. Over 80% efficient.
  Relatively expensive.
* ★ Brushless.
  No brush wear, hence no arcing and acoustical noise. Compatible with solid-state circuits. Independent speed and torque.
  High torque-to-inertia.
  Bi-directional. No tacho required.
  Medium efficiency.
  Relatively expensive.

also becoming important for robotic applications. The interface between the data bus and the stepper is relatively simple and inexpensive and these motors provide extremely accurate positioning, much more so than can be achieved when using a dc motor with an encoder. Errors are only a few per cent of a step therefore, with a 1.8 degree motor, the error is only minutes per step. Furthermore, errors do not accumulate.

But not all applications suit the characteristics of a stepper, and especially when an application requires power only. As a general rule, high-power and high-torque are not for steppers. However an overall stepper system is simpler and less costly than that of the corresponding servo system plus amplifier, and a stepper is usually more stable and rigid at rest. However, they are unable to accelerate as quickly as a dc motor and have reduced limits on torque and speed.

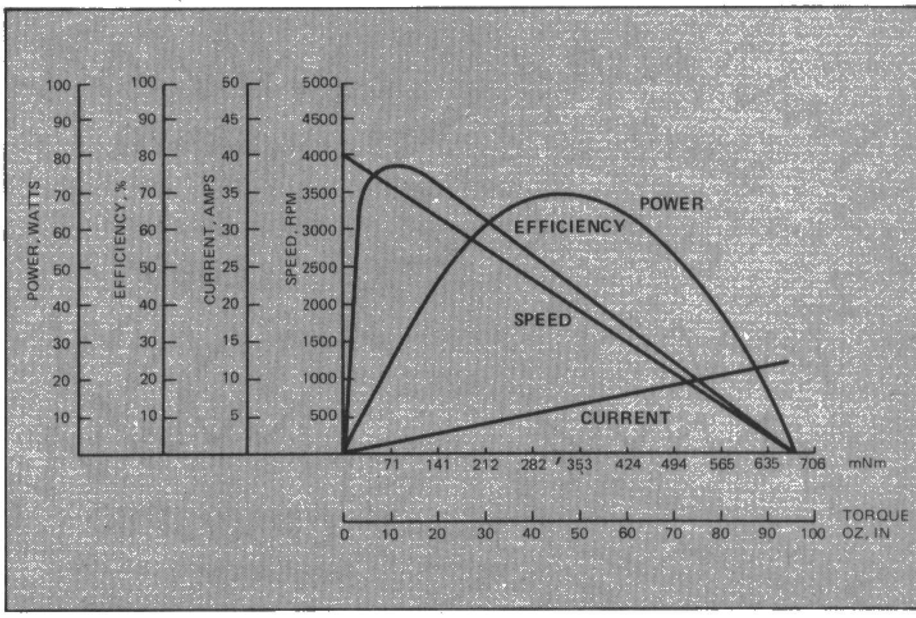


Figure 3 (above and below) shows typical performance curves of a brushless dc motor.

TYPICAL SPECIFICATIONS (25°C)–	
NO LOAD SPEED	4000 RPM
PEAK TORQUE	636 mNm (9 OZ. IN)
PEAK POWER	60 WATTS
NO. OF PHASES	3
NO. OF POLES	4
TORQUE CONSTANT	55 mNm/A (7.8 OZ. IN/A)
VOLTAGE CONSTANT	0.055 VOLTS/RAD/SEC (5.8 VOLTS/1000 RPM)
APPLIED VOLTAGE	24V DC
INDUCTANCE LINE TO LINE (@ 1000Hz)	1.4 mH
COIL RESISTANCE LINE TO LINE	1.3 OHMS
THERMAL RESISTANCE*	12°C/WATT
INERTIA	0.42 GM^2 (0.06 OZ. IN/SEC^2)
WEIGHT	0.55KG (1.2 LBS)

* STATIC: STATOR ROTOR KIT, NO HEAT SINK

DC MOTOR OPTIONS

For low power, computer associated equipment the low contact resistance between commutator and brushes of a conventional iron-core rotor is attractive because of its good start up characteristics. Power consumption is low and, therefore, this type of motor suits battery operated products.

An iron-less motor has low rotor inertia, and low inductance. Furthermore, it doesn't cog. In some designs the commutator, which has a disc and noble metal brush arrangement, is equipped with zener diodes or capacitors to increase the motor's performance capability and to give it extended life. These components provide protection against voltage spikes and excess current overloads.

"Controllers are able to provide more precise positioning information than most steppers are able to deliver".

Brushless motors, which employ semiconductor devices (Hall generators) instead of a conventional commutator with brushes, have their rare earth magnets mounted on the motor shaft. The motor armature is on the stator, which is opposite to a normal PM motor. This overall arrangement brings the advantages of a lighter rotor with less inertia, all heat being dissipated in the stator from where it is easily removed. Moreover, the design takes away any problems of mechanical wear and energy dissipation at the commutator, and thermal stresses on the bearings are reduced.

The performance characteristics of brushless motors, **Figure 3,** are similar to those of the PM motor: motor speed is proportional to supply voltage and speed deviates linearly with torque. Changes in either the supply voltage or current will vary the speed of the motor and its torque capability. The starting torque may be as much as four times the rated continuous torque.

EXTRA TORQUE THROUGH GEARS

The transmitted torque of a motor may be increased by driving through reduction gearing. Torque is proportional to the reduction ratio – less gearing efficiency – and output speed is reduced accordingly.

Gearing will also improve the positional resolution of a motor. It is especially significant for steppers because the gearing reduction reduces the reflected inertia of the load on the motor. For instance, if load inertia is J, then reflected inertia on the motor is — where n is the reduction ratio.

LOAD ASSESSMENT

Ultimately, the programme for any motor controller will be based on information relating to the job to be performed. The relationship that has to be considered in selecting a controller is the load that acts on the motor and the motor's characteristics., Note the speed/torque relationships between the stepper and dc motors of similar input power in **Figure 4.**

There are three common motor loading conditions: frictional, inertial, and lead-screw applications. Although we are considering the load on a stepper, don't overlook the fact that a servo may deliver a smoother and better performance in some situations. For lead-screw duty, the pitch of the thread, the screw's diameter and length, as well as the type of thread, all affect motor loading.

In relating controller to motor duty ensure that there is sufficient capability to handle load travel per step of motor rotation, provided different motor speeds for acceleration, and also to take care of slewing speed. As far as resolution and accuracy are concerned, they are primarily functions of the motor rather than the controller. Generally, controllers are able to provide more precise positioning information than most steppers are able to deliver under actual operating conditions.

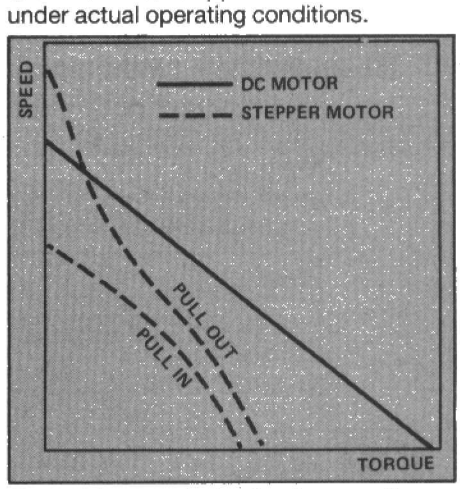


Figure 4 is a graph showing the speed/torque curves for stepper and dc motors of similar input power.

BUILD YOUR OWN . . .

LOW COST ROBOTS

Richard Sargent, in the first of a series of articles, describes the construction of robots using LEGO® building bricks.

*LEGO is a registered trade mark

As the cost of chips and computers steadily falls (notwithstanding temporary hiccups in the TTL market), the wide disparity between the price of the micro itself and its real-world interface and hardware becomes all too apparent. However, there is a way to start experimenting with robots and machines which requires only a small initial outlay of cash.

The modular building systems are perfectly capable of being made into small and useful robots, and this series will use LEGO bricks to prove the point. The LEGO system has come a long way since its early days as a building block toy, and can be quickly used to construct highly accurate pieces of equipment. This series will describe four robotic "things" which can be built from a reasonably modest selection of engineering LEGO. Once you have experimented with a particular robot, it can be dismantled and the pieces used on the next project.

The series will progress from simple interfacing, software and models to fully operational pieces of equipment, which will include a floor-roaming BUGGY and a multi-axis ARM.

MOTORS

Most small demonstration robots rely upon low-geared motors and solenoids to drive them. The series will avoid solenoids, since they are difficult to obtain, tricky to fit and consume a great deal of power. All movement will be instigated by the common low-voltage DC motor, or which there are a great many to be found cheaply, either new or second-hand. The preferred motor is the 4.5V LEGO motor, which runs well from three "C" size batteries. There is scope in the design of the robots to use some motors other than the LEGO ones, and at voltages other than 4.5V. Wheat is essential though, is that the motors are low-geared, and the combined use of mechanical gearing and software speed control can usually bring about the desired motion.

DEGREES OF FREEDOM

Most robots have a multitude of motors which simultaneously control their moving parts, and this is especially so in the case of the arm robots. The point where one rigid part is fixed to another is called a joint or axis, and each axis is said to give the robot one degree of freedom because it allows the parts fixed at the joint to move in a certain way. An interesting arm robot may have six or seven degrees of freedom, but remember, each axis requires a motor to drive it. Positioning six or seven motors represents a fair engineering problem, and to the home constructor, a significant financial outlay.

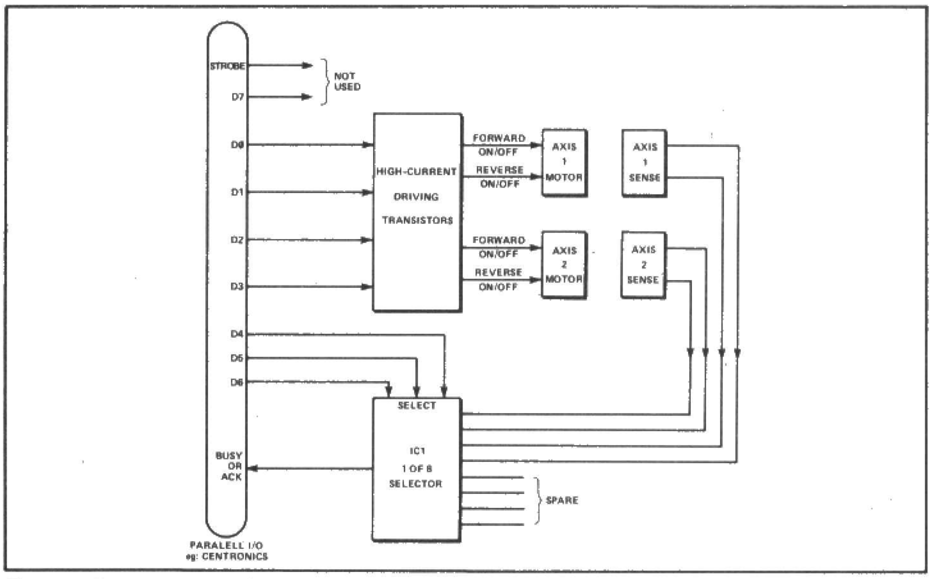


Figure 1. Block diagram of the robot's electronics.

The robots in this series will require between two and four motors, and to help reduce costs the possibility of mechanical interface between low cost motors (such as those found in toy cars from the Far East) and Technical LEGO will be discussed, and methods described.

ROBOTIC WALL BUILDING

The first project concerns a small mobile robot which is blessed with two degrees of freedom and has a liking for building and rebuilding brick walls. You will need two LEGO motors and various other LEGO bits and pieces to build it, and the computer will be able to control it by using BASIC through a Centronics or similar input/output port. The block diagram of the electronics is given in **Figure 1.** Before giving the parts list for the LEGO Wall-builder, some further general comments on LEGO bricks as a building medium might prove useful.

GETTING IT TOGETHER

A good starting point is to obtain a LEGO Catalogue for 1984*, which, apart from having a wealth of information on kits a million miles removed from the subject of electronics, has all the reference numbers of all the (robotically) important LEGO kits. It also has a useful order form which will allow you to get those ever-so-important pieces of which no kit ever seems to have enough. **Table 1** describes some of the LEGO kits which contain accessories rather than models. It should not be regarded as a parts list for the projects, but merely as a guide to what's available. These kits are available from toy shops, departmental stores and mail-order companies. **Table 2** lists some of the useful spares which can be ordered direct from LEGO UK.

It is anticipated that constructors will be inventive and improvise with Lego and other suitable materials that come to hand, in the best British tradition. Next month's Wall Builder will require two geared motors of the type found in kit 107, some type-1230 chain link and some assorted gears and beams. Full constructional details will be provided to enable you and your robot to drive yourselves up the wall!

TABLE 1. Some useful kits.

107	Geared motor and accessories.
8700	Ungeared motor and accessories.
8050	STARTER SET with ungeared motor.
744	BASIC SET with geared motor.
872	Reduction gearing set.
8710	Technical beams.
8750	Rails.
843	Base plates (also 840, 302).
820	Small plates (also 822).
830	Bricks

TABLE 2. Mail-order spares.

1206	Round bricks.
1207	Turntable.
1217	Technic beams & plates.
1221	Technic long beams.
1225	Assorted axles.
1227	Assorted gears, pulleys.
1228	Rack, specialised bits.
1229	70 links of narrow chain.
1230	54 links of broad chain.
1231	Two 82mm dia. wheels.
1232	Beam pivots, axle joiners etc.

Available from: Spares Service, LEGO UK, Wrexham, Clwyd LL13 7TQ.

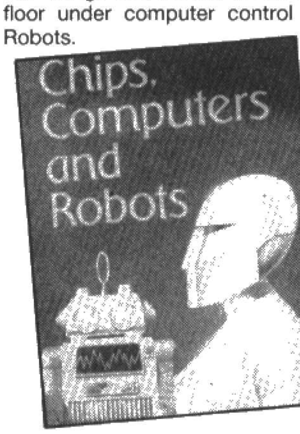
BOOK REVIEWS

Robotics books are spawning like rabbits in spring. We review three practical tomes.

How to design and build your own custom robot

by David L. Heiserman
Tab Books Inc.
(Distributed by Foulsham and Co.)
Paperback 462 pages £12.60
First published 1982

This book is not, I am relieved to say, the type of American text which gives page after page of engineering details about axles and gearing and other boring stuff with a view to creating one specific wheeled-robot which runs from an out-dated computer. On the contrary, the book includes plans for a variety of microprocessor robot systems, with thorough explanations of the electronic hardware which make up the various sensing and driving elements of the average robot. Mr. Heiserman is at pains to point out though, that what is commonly called "ROBOT" is in fact a "PARABOT". A parabot is a machine that lacks an element of autonomy or self-determination – in other words it is controlled directly by a human operator or indirectly by means of a prescribed program. So the book describes the building of parabots, and gives a little information on the vastly more complicated subject of true robots. Personally, I don't care for this American habit of inventing new names for things, but if you can forgive Mr. Heiserman for throwing AI (Artificial Intelligence) out of the window in favour of EAMI (Evolutionary Adaptive Machine Intelligence) then you will get on well. To be fair, he does argue his case and dispense with these matters of nomenclature in the introductory chapter. To me, little things that trundle around the floor under computer control are Robots.



The information contained in the book is substantial, but the author does assume that the reader has a background in basic electronics and an understanding of certain fundamental points. For example, source-code listings (8080/8085 & Z80 code) are given towards the end of the book to illustrate certain routines, and although they are accompanied by good flow-charts, they assume the reader has a sound grounding in machine code language.

The electronics are presented in a clear and modular fashion, and use common integrated circuits, transistors and other components. The fact that the book is an American publication shouldn't matter too much as mains voltages and UHF modulation don't impinge on the subject matter, but the British suppliers thoughtfully provide a little pamphlet entitled "Using American handbooks in Britain" which deals with colour-codes, wire gauges and screw sizes etc and this should ensure that the reader doesn't get stuck when ordering supplies.

Equations accompany the circuit diagrams, enabling you to work out, for example, the relationship between the motors and the batteries that you have chosen to use. Since you won't be running your floor robot from penlight cells, the early chapters of the book deal with selecting your batteries (Lead-Acid types probably) and the all-important battery charger. Electronic charging systems are dealt with, as are low power detection systems and, naturally, a system which sends the robot hunting for a recharge station when it feels weak. This, I understand, is called a "Nest search". Motor driving and stall-sensing circuits are dealt with in great detail and you will find a wealth of information on speed control in particular. The theory of magnetic, optical and electro-mechanical sensing schemes are covered in detail, but apart from a few constructional hints, the user is left to devise his own plans for jointed arms and grasping hands. Range-finding, speech synthesis and voice-recognition circuits are notably absent, and this is perhaps the only indication that the book is some four years old, being now in its second printing.

Details are given for connecting the robot to a microprocessor, ROM and RAM, and the circuits, pin designations and ideas are very sound, except that the RAM memory chips specified are somewhat dated. At today's prices it would probably be cost-effective to tuck a converted 16K Spectrum into the beast, and merely concentrate on building the I/O circuits. If you wish to build a substantial floor-roaming robot, this book will give you a flying start.

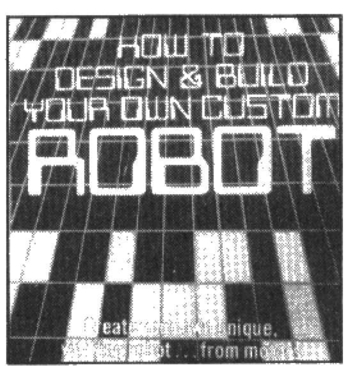
Chips, Computers and Robots

by Judy Allen
Puffin Books
Paperback 95 pages 95p
First published 1982

This strange little book, revised in 1984, is a brief introduction to chips, computers, robots "and beyond". Puffin Books would have the reader believe that it is a "straightforward, fascinating explanation of microtechnology". Straightforward it may be, but fascinating it is not.

I am somewhat at a loss to decide who the book is intended for. Puffins are usually aimed at the lower age ranges, but the author certainly doesn't use a style one would expect when communicating to children – the chapter on chip-manufacture is a case in point. The line drawings, which occupy about 40% of the book are uninspiring and not always relevant.

However, Judy Allen appears to cover a lot of ground, without actually revealing too much. The omission of any discussion on the home micro, the school micro and the tremendous upsurge in hobby microelectronics generally is a major flaw and might suggest that the origins of the book lie in the 70's.



The section on robots is really nothing more than a catalogue of the various types of machine, but here, too, the text is strangely flawed, and it is sometimes difficult to separate the fact from the fiction. The household robot designed by Quasar Industries in New York in 1977, which received much publicity at the time, was, if I recall correctly, nothing more than an elaborate publicity stunt. This is not made clear when it appears in chapter 31 "The housework robot".

As light reading, the book is innocuous enough, and may well serve a useful purpose in whetting your appetite for further study. As an explanation of microtechnology, it leaves a lot to be desired.

What robots can do and how they work

by Tony Potter and Ivor Guild
Usborne New Technology Series
Paperback 48 pages £1.99
First published 1983

Usborne issue a large range of colourful paperback books on technology and computer-related subjects and this is certainly one of their best. It is up-to-date, interesting and unpatronising in its approach to the subject.

The book begins on a sound note with the information that the word "robots" was first used some 60 years ago by a Czechoslovakian playwright, and is not an invention of Hollywood. The limitations of present-day robots are explained, and the problems of designing that much sought-after beast, the household robot, are carefully pointed out. All the robots that youngsters are likely to meet in schools are mentioned, a few in some detail, and the industrial, military and scientific robots get their fair share of attention too. NASA is not forgotten: the topical Remote Manipulator System (that's the Space shuttle's arm, by the way) is well highlighted but, sadly, the more sophisticated Mars Lander is mentioned only in passing.

Part of the Usborne style is to give coherent slices of information in two-page spreads, and the pages on "How arm robots work" and "Types of arm robot" are particularly well presented. To have "degrees of freedom" and, "working envelopes" so clearly explained is a very welcome feature.

Driving, sensing and computer control are explained, and the difficult concepts of artificial intelligence and related topics (pattern recognition for example) are treated carefully.

Usborne books are not for armchair readers. There are always suggestions for "things to do". This book has a number of projects. There is a zero-cost pneumatic hand gripper made from everyday household objects, or for the more adventurous there are plans for a simple floor turtle which can be interfaced to a micro. Full constructional details are given, and there is a BASIC turtle-driving program provided for the BBC, VIC, ZX81 and Spectrum home computers.

This book is available on its own or as part of a 144 page "New Technology" compendium and it gives a very complete introduction to the exciting world of Robotics. I can thoroughly recommend it.